A Probabilistic Framework for Information Modelling and Retrieval Based on User Annotations on Digital Objects

Vom Fachbereich Ingenieurwissenschaften
der Universität Duisburg-Essen
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
genehmigte Dissertation

von

Diplom-Informatiker
Ingo Peter August Frommholz

aus Bochum-Wattenscheid

Referent: Prof. Dr.-Ing. Norbert Fuhr
Korreferentin: Prof. Dr. Maristella Agosti
Tag der mündlichen Prüfung: 21. Oktober 2008

Für Damaris.

Abstract

Annotations are a means to make critical remarks, to explain and comment things, to add notes and give opinions, and to relate objects. Nowadays, they can be found in digital libraries and collaboratories, for example as a building block for scientific discussion on the one hand or as private notes on the other. We further find them in product reviews, scientific databases and many "Web 2.0" applications; even well-established concepts like emails can be regarded as annotations in a certain sense. Digital annotations can be (textual) comments, markings (i.e. highlighted parts) and references to other documents or document parts. Since annotations convey information which is potentially important to satisfy a user's information need, this thesis tries to answer the question of how to exploit annotations for information retrieval. It gives a first answer to the question if retrieval effectiveness can be improved with annotations.

A survey of the "annotation universe" reveals some facets of annotations; for example, they can be content level annotations (extending the content of the annotation object) or meta level ones (saying something about the annotated object). Besides the annotations themselves, other objects created during the process of annotation can be interesting for retrieval, these being the annotated fragments. These objects are integrated into an object-oriented model comprising digital objects such as structured documents and annotations as well as fragments. In this model, the different relationships among the various objects are reflected. From this model, the basic data structure for annotation-based retrieval, the structured annotation hypertext, is derived.

In order to thoroughly exploit the information contained in structured annotation hypertexts, a probabilistic, object-oriented logical framework called POLAR is introduced. In POLAR, structured annotation hypertexts can be modelled by means of probabilistic propositions and four-valued logics. POLAR allows for specifying several relationships among annotations and annotated (sub)parts or fragments. Queries can be posed to extract the knowledge contained in structured annotation hypertexts. POLAR supports annotation-based retrieval, i.e. document and discussion search, by applying an augmentation strategy (knowledge augmentation, propagating propositions from subcontexts like annotations, or relevance augmentation, where retrieval status values are propagated) in conjunction with probabilistic inference, where $P(d \rightarrow q)$, the probability that a document d implies a query q, is estimated. POLAR's semantics is based on possible worlds and accessibility relations. It is implemented on top of four-valued probabilistic Datalog.

POLAR's core retrieval functionality, knowledge augmentation with probabilistic inference, is evaluated for discussion and document search. The experiments show that all relevant POLAR objects, merged annotation targets, fragments and content annotations, are able to increase retrieval effectiveness when used as a context for discussion or document search. Additional experiments reveal that we can determine the polarity of annotations with an accuracy of around 80%.

Acknowledgements

I would like to take this opportunity to thank those who accompanied me on the long way throughout the time this thesis was created, who supported me in several ways and who showed interest in my work.

I thank my former and current colleagues at Fraunhofer IPSI in Darmstadt and the Information Systems group at the University of Duisburg-Essen, especially Holger Brocks, André Everts, Marcello L'Abbate, Adelheit Stein, Matthias Hemmje, Sascha Kriewel and Claus-Peter Klas. They always found the time for discussion, to give technical support or just to listen. Special thanks go to Henrik Nottelmann, a brilliant nice guy who left us much too early, to Erich Neuhold, who was involved in my work when he was institute director at IPSI, and Piklu Gupta, a native English and a near-native German speaker (and, besides, a nice guy), who helped to translate even complicated German sentences into English. I also thank Marc Lechtenfeld for his fantastic master thesis on machine-learning methods to determine the polarity of annotations, Dennis Korbar, who helped me by providing the infrastructure to create the ZDNet testbed, and Ray Larson for reading an early version of this thesis.

Ulrich Thiel was the one who mentored me during my time at Fraunhofer IPSI. He showed me the "other side of IR", namely the cognitive, more user-oriented one. Ulrich's comments sometimes gave me a very hard time, but made me learn a lot.

Thomas Rölleke made the heart of this work possible by providing his superb HySpirit framework. Without him, none of the proposed framework could actually be executed. Thanks for good advice, a nice afternoon on a sailing boat and your patience for answering many questions. And of course thanks for POOL.

During a visit to Padua, I had the opportunity for good and fruitful discussions with Maristella Agosti and Nicola Ferro. Their collaboration enriched my work significantly. I'd like to thank them for good advice, the nice time I had with them, for the good collaboration in DELOS and for their interest in my work.

Especially I'd like to thank Norbert Fuhr. He is the person mainly involved in my work. His inspiration, his deep knowledge and his support paved the way to make this thesis possible. He was also the one giving me the opportunity to continue the work started in Darmstadt when I began working at his chair in Duisburg.

Finally, very hearty thanks go to my family and especially my wife Damaris. She is the one who was suffering most when I was writing up this thesis, and her infinite patience cannot be measured.

Thank you.

Ingo Frommholz
Darmstadt/Duisburg, November 2008

Contents

1

Introduction

Imagine that you want to buy a new smart phone and wonder whether your preferred GPS-based navigation software runs on the desired device. Using common search engines on the Web, you find evaluation reports about the phone and its technical features, but unfortunately, these reports do not consider your specific requirement. But some of these Web sites allow for writing comments on the article, and you write a comment on one evaluation report, asking

> `Does anyone have experience with the XY navigation software and the ABC mobile phone?`

After waiting a few hours, you revisit the page again and find that, triggered by your question, a whole discussion started among the users how to make the navigation software run on the smart phone. You are very satisfied with the outcome of the discussion, buy the desired smart phone, follow the instructions found in the discussion, and after the first cruise with the running navigation software on your new smart phone, you write another comment to the forum:

> `Thanks, this helped a lot!  :-)`

Later, another user somewhere in the world has the same problem you had. She types appropriate keywords into the search engine, and this search engine ranks the evaluation report mentioned above on the first position, additionally hinting her that she should also consider the comments to find the information she needs. She navigates to the comments, and after only a few minutes she found exactly what she wanted. Reading your last comment, she finds that the approach described in the discussion thread is really worth trying.

Imagine another scenario. You are a film scientist and want to write an article about political censorship. For your research, you have access to a repository of scanned documents about movies in the last century. In the search interface of the system, you type the query terms "political censorship", and the system returns some results. While examining these, you find a censorship document about a film which gives morality issues as the censorship reason. You wonder what this document has to do with your research, when you discover an annotation saying:

> `I think the film was censored for political reasons.  Since political censorship was not allowed at that time, they just had to find other reasons.`

This comment, coming from a scholar working at another institute, makes you curious. You discover that this interpretation is not the only annotation about the document, but that a

whole discussion thread was started with lots of agreement and disagreement. You read the annotations in the discussion thread and gain valuable insights and new references for your own article.

Now consider another case. You want to write a master thesis about digital libraries, and to prepare for this thesis, you need a good and easy to understand introduction to the field. You go to the Web site of a book seller which offers some functionality to search within the book titles. You type "digital libraries introduction" as a query and find a few books which are supposed to be introductions. Since you only want to read one of the books, you need to find out which one of them is really introductory and easy to understand. Fortunately, the system also presents user reviews, and for one book, you find two important comments:

```
This book is a good introduction!

It was easy to read and helped me prepare for my exams.
```

Although this book was not in the first position of the ranking produced for your search, you choose it due to these comments.

A fourth and last scenario. You are writing a PhD thesis and you want to find a certain paper, but you forgot the title and do not remember the author. All you remember is this passage in the paper which you found very interesting. You remember marking it with a green marker and writing a comment on the margin: "Good idea, should include it in my model!". Looking at the pile of papers you read throughout the last year, you would give everything for a hint where to find this annotated passage again! So you make your way through the pile, and finally, after having examined dozens of papers for this single annotated fragment, you find it, re-read the important parts of the paper, and integrate its basic idea into your model.

All scenarios, which are fictitious but could have happened at any time, have something in common: user annotations are exploited to satisfy an information need. In the first scenario, main documents like the evaluation report could not answer the specific question if the navigation software runs on the desired smart phone; later, the answer could be found in the comments. In this scenario, we hinted at an imaginary search engine which "knows" that the desired information is in the annotation-based discussion, and could therefore easily point the following user with the same problem to it. In the second scenario, which comes from the context of the COLLATE project presented later, a document is associated with the information need ("political censorship") by way of annotations, and can therefore be deemed relevant. Furthermore, since the main documents the system deals with are scans, and search engines usually operate on textual descriptions of their material, the annotations may be one of the few sources a search engine can exploit to present the user relevant documents. The third scenario shows the usefulness of annotations saying something *about* a document. From all the books offered by the system, the one was chosen which seems to best fit the additional requirement of being a good and readable introduction. Finally, in the last scenario, a document was found (again) due to an annotated fragment.

These introducing scenarios shall motivate the main focus of this thesis: to use annotations as an additional source for information retrieval to satisfy certain information needs. The examples above are only few of the possible ones involving annotations. In fact, the act of annotation is many centuries old and has a long tradition and many applications in the non-digital world. But in recent years, annotations have entered the digital world more and more. We find them in commercial office tools like Word or OpenOffice (where people can insert comments into the text or highlight important parts), in digital libraries which let users annotate (and through this, interpret) the material at hand, and in form of product reviews and discussion forums

attached to some documents on the Web. From a certain viewpoint, even well-known technologies like email can be regarded as digital annotation. This recent development lets novel search and retrieval approaches be able to access the rich additional source of information contained in annotations in order to perform document or discussion search. While traditionally in information retrieval, documents are regarded as an atomic single unit, we are now able to see these documents embedded in the context established by annotations. To fully exploit this context for retrieval, it is not sufficient to define yet another retrieval or term indexing function; this would probably miss information contained in the annotations or about them which is valuable to satisfy specific sophisticated information needs. Annotations can be employed to extend the knowledge in a document, but we might also want to use non-topical information, for example in cases annotations contain statements about the quality of the annotated object (think of book reviews for instance). Or we want to search for documents through annotations, for example when we recorded an idea in an annotation and now want to find the passage in a book again which was the source of this idea. We see that there are a number of information needs and search strategies thinkable which can potentially be satisfied and supported when considering the annotation context in a suitable way. What we need is a flexible framework which can handle these needs and support these strategies by exploiting as much of the information as possible contained in the annotation context. But before such a framework can be designed, it is important to study contemporary annotation systems in order to extract important facets of annotations and create a model which reflects all important objects playing a role in an annotation scenario. And of course we need to answer the question if such a framework, which incorporates annotations into the retrieval process, can potentially improve the retrieval quality.

Some of the basic questions discussed in this thesis therefore are: What is contained in the annotation context? Which objects can we identify conveying information exploitable for retrieval? What is "annotation-based retrieval", actually? How can we model annotations and related objects to make them satisfy our information needs? And does it really perform better than traditional retrieval approaches? To provide answers to these questions, the thesis is structured into the three parts shown in Figure 1.1. The main question this thesis tries to answer is: How and how effective can we exploit annotations for information retrieval? The three parts of the thesis address this question.

Before we can discuss how to use the context established by annotations for information retrieval, we need to shed some light on the question what this annotation context actually is. What objects do we find there, and how are they related? Part I discusses these questions. First, the notion of "annotation" and contemporary annotation systems and studies are investigated in Chapter 2. From this investigation, some important facets of annotations are derived. In Chapter 3, we pick up the discussion in Chapter 2 and formulate an object model of the annotation universe, which results in the definition of the structured annotation hypertext on the instance level, combining structured objects with annotations. This model comprises important concepts for annotation-based retrieval and can be used as a base data structure to develop annotation-based retrieval approaches.

Part II builds upon the output of the first part. After defining what the annotation context consists of, namely structured annotation hypertexts, this part asks for the "How?" in the main question about exploiting annotations for retrieval. Since the annotation context is quite complex and full of information on various levels, a probabilistic, logic-based framework called POLAR is introduced in Chapter 4. This goes beyond defining a simple context-based indexing or retrieval function (although such a retrieval function, realised by means of what we call knowledge augmentation and probabilistic inference, is the core of the framework).

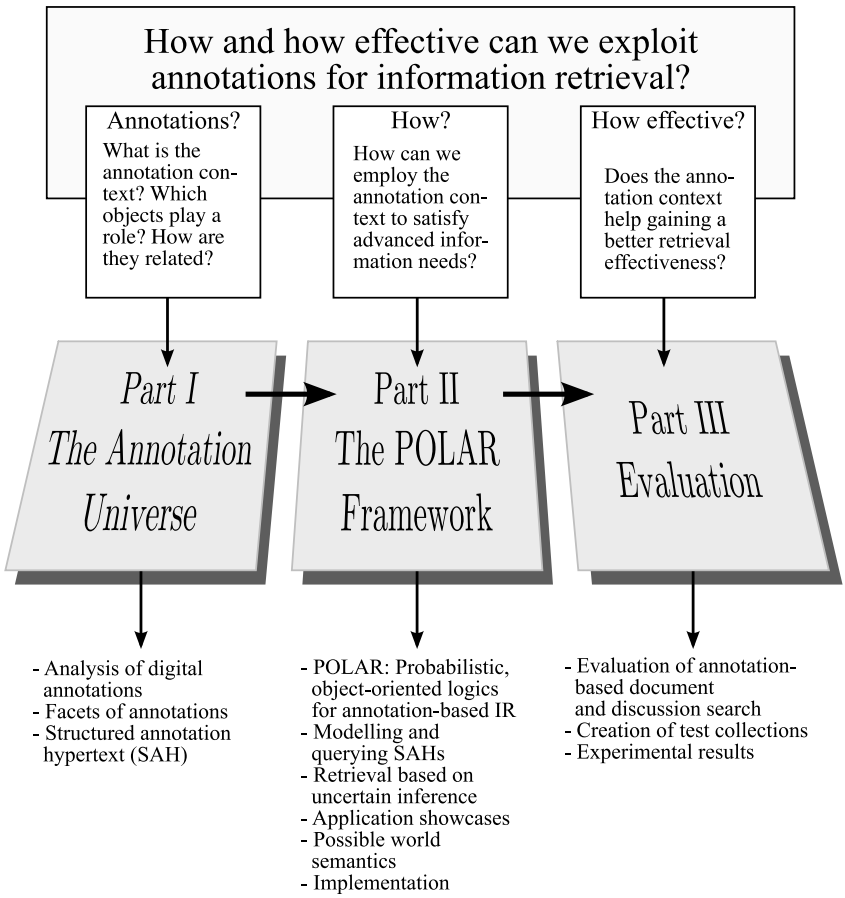


Figure 1.1: Structure of the thesis

POLAR is an extension of another logic-based framework called POOL, which is targeted at modelling structured documents (but no annotations). POLAR is expressive enough to represent structured annotation hypertexts and to support advanced information needs, which also involves non-topical information. It is supposed to be applied in many annotation-based retrieval scenarios. To do so, POLAR models and queries structured annotation hypertexts, and combines structured with annotation-based retrieval. Besides providing complex querying mechanisms to the underlying knowledge base, POLAR's main features are knowledge and relevance augmentation, which exploit (among others) the annotation context, and retrieval based on the estimation of the implication probability $P(d \rightarrow q)$ that a document d implies a query q. To illustrate that POLAR is able to support sophisticated information needs, some example application showcases are given. POLAR is a logic-based framework, which means that besides its syntax, also its semantics have to be defined; both are done in Chapter 5. The semantics are based on possible worlds and Kripke structures and extend those defined for the POOL framework in order to handle annotations and their various peculiarities. Chapter 6 discusses the implementation of POLAR, which is done by translating POLAR programs into four-valued probabilistic Datalog (FVPD). FVPD programs can then be executed with engines like HySpirit.

Part III discusses the "how effective" w.r.t. exploiting the annotation context for retrieval. It answers the question if the incorporation of the annotation context into retrieval helps to gain

better effectiveness, at least for two manifestations of annotation-based IR, document and discussion search. It describes the evaluation of POLAR's main approach, which is retrieval based on estimating the implication probability and using knowledge augmentation. In Chapter 7, the test collections (W3C Email discussion lists and ZDNet News) are introduced. Chapter 8 discusses experiments performed with these collections and their results. These experiments shall answer the question if annotations and their related objects can be used to improve retrieval effectiveness. A further series of experiments deals with another question: How accurately can we determine if an annotation as a reply to another annotation is positive or negative?

Chapter 9 concludes this work and provides an outlook on future work.

The main contributions of this thesis are:

- an annotation model (the structured annotation hypertext) reflecting the results of annotation research and projects, with the focus on annotation-based IR, as the output of Part I;
- the POLAR framework to model structured annotation hypertexts and exploit them for annotation-based IR in order to satisfy advanced information needs (output of Part II);
- the evaluation of some POLAR concepts introduced in Part II, in particular knowledge augmentation with probabilistic inference, for annotation-based document and discussion search. This is presented in Part III.

While the scenarios above give a first clue about annotations and their possible role in information retrieval, we commence this thesis with an overview of what we call "the annotation universe" in the next chapter. This study shall define and reveal the nature of annotations, its facets and components, and how they are applied in current systems.

Part I

The Annotation Universe

2

The Annotation Universe – Applications, Facets and Properties

May your journey be free of incident.

(Vulcan proverb, "Star Trek IV: The Voyage Home")

In order to design annotation-based retrieval functions, we first have to understand the concept of annotation – its application, its facets and its properties. Therefore, this chapter contains a study of annotations, of the applications employing annotations, of its facets and properties, and the identification of further important entities involved in the annotation process. To understand what a "digital annotation" actually is, we do not have to start from scratch, but can rely on many studies and experiences with digital annotations. However, this chapter should not be seen as yet another thorough study on these issues, as there have been many such studies before, but it is directed towards the goal of exploiting annotations for information retrieval. The chapter starts with a description of digital annotations, their definition and application in several fields and systems. It continues with a reflection of what annotations are used for. After that, different facets of annotations are discussed.

2.1 Digital Annotations

2.1.1 Definition and Usage

Presumably everyone reading this thesis has used annotations at some time. We meet them in form of small notes and comments written in the margin of an article we need to read and understand for a certain purpose, or as highlight annotations to mark important parts of a paper. Annotations can also be references pointing to another passage in the text (intra-document) or to another document (inter-document). Annotations in form of interpretations are used, e.g., to understand classical literature[1], legal texts, or religious books like the Bible or the Talmud (Fraenkel and Klein, 1999). These examples also show that the practise of annotation goes back to (at least) the Middle Ages; in fact, annotations can be found on

[1] In Germany, for example, the famous "Königs Erläuterungen" provide interpretations of classical literature like Goethe's "Faust" (which is an essential information source for pupils preparing their exams).

historical documents, not only in the literary or theological field, but also for administrative and legal practise (Agosti et al., 2007a).

The concept of annotation is best introduced by looking at common definitions. Webster's New Encyclopedic Dictionary defines the verb "to annotate" as

> to make or furnish with critical or explanatory notes or comments

and "annotation" as

> a note of comment and explanation

Hornby et al. (1976) elaborate the definition of the verb "to annotate" as

> to add notes (to a book, etc) explaining difficulties, giving opinions, etc.

We see with these definitions that annotations are related to notes and comments, and that their usage might be, among others, to explain things or give (critical) opinions. Furthermore, we see that annotations are a kind of add-on to the object they belong to.

As Agosti et al. (2007a) point out, there are many other terms related with annotations, like footnote[2], gloss (an explanation or interpretation of a word), jotting (a brief note), observation (comment or remark about the fact learnt from observation), postil (comment or marginal note), record (e.g., information about facts or events) or scholium (commentary or annotation, especially on classical text). Another kind of annotation are highlight annotations which mark important passages of a text (either by underlining them, drawing a circle around them or marking them with a highlighter).

The idea of annotations in non-paper form goes back to the year 1945, when Vannevar Bush presented his idea of the revolutionary "memex" device, which is always regarded as a vision of today's digital libraries (Bush, 1945). Within the memex device, it should be possible to record comments and notes and attach them to the objects they comment. In a way, the idea of "digital annotations" was born even before digital computers started their successful story. In the Eighties, Halasz (1988) mentioned the creation of annotations as one kind of activity to support collaborative work. In the last roughly 15 years, with the emergence of digital libraries and the Web, efforts were made to transfer the concept of annotation, as we know it from the paper world, to the digital world. In fact, there are certain benefits in introducing annotations to the digital world. Once annotations are machine-processable, they can easily be shared and reused. We can make our notes and comments available from any location or store them on our laptop or mobile device. Annotations may be one of the last reasons why we print documents instead of reading them electronically. But to make our paper annotations available everywhere, we usually need to carry a pile of documents around. And, finally, digital annotations can be integrated into the processing of documents and in operations applied to them, which may be, for example, indexing and retrieval. Digital annotations thus can serve purposes which paper annotation cannot. So we see there is a certain value in making annotations digital.

The act of annotation always serves a special purpose, although the way an annotation is used may change over time. Ovsiannikov et al. (1999) identify four primary uses of annotations, which are: to *remember*, to *think*, to *clarify* and to *share*.

Annotations help *remember* the main content of a document. As an example, highlighted and annotated passages indicate important parts of a text. Shipman et al. (2003) assign such

[2] a note printed at the bottom of a page

passages an *emphasise value*, which is a heuristic measure; more focused annotations (focused regarding the passage they annotate) have a higher emphasise value as do interpretive annotations which have a comment attached. Annotations with only highlighting have a lower emphasise value.

Active reading is the combination of reading with (critical) *thinking* and learning and is a fundamental activity of knowledge workers (Schilit et al., 1998a). Active reading usually is accompanied by annotations. Giving critical remarks, asking questions and creating notes reflecting the opinion of the reader, as well as recording thoughts and ideas (in the context of the document) are facilitated by means of annotations.

Annotations containing interpretations can help a reader to *clarify* ideas in the text which are hard to grasp. Furthermore, as another variant of active reading, reshaping parts of the document content in one's own words can be a good means to understand this content, and to locate points which need more clarification. Summaries, which can also occur in form of annotations, might help to bring the single ideas and assertions presented in a document into a global context.

The *sharing* of annotations is a very important means to foster collaborative work or the reuse and communication of previous ideas and interpretations. With shared annotations, users can discuss topics or ideas, give feedback, or collaboratively interpret the given material. The sharing of annotations is possible by defining the *scope of an annotation*; usually, this scope can be *private* (only visible to the author of the annotation), *shared* (visible to the members of the same group) and *public* (visible to everyone). Shared and public annotations are the building block for annotation-based discussion (Brocks et al., 2002; Bernheim Brush, 2002).

2.1.2 Annotations in Digital Libraries and Collaboratories

2.1.2.1 Role of Annotations in Digital Libraries and Collaboratories

One of the main areas where annotations are applied are digital libraries and collaboratories. *Digital libraries* (DLs) are not only the digital versions of traditional libraries, but offer means going beyond mere presentation of the content stored in a digital repository. Two definitions of digital libraries, coming from two different directions and thus focusing on different aspects, point to this fact. The introduction to the first issue of the *International Journal on Digital Libraries* (cited in Fuhr et al. (2001)) expresses a more computer science oriented view on DLs:

> Digital libraries are concerned with the creation and management of information resources, the movement of information across global networks and the effective use of this information by a wide range of users.

A slightly different viewpoint on DLs comes from librarians themselves:

> Digital libraries are organisations that provide the resources, including the specialised stuff, to select, structure, offer intellectual access to, interpret, distribute, preserve the integrity of, and ensure the persistence over time of collections of digital works so that they are readily and economically available for use by a defined community or set of communities. (Digital Library Federation (DLF), 1998, cited in Fuhr et al. (2001))

As discussed in Agosti et al. (2004), annotations can potentially support many functions DLs should provide. Among them are the creation of new documents: first, annotations are new information resources themselves; second, annotations convey the content which is then compiled

into a new document. When creating new annotations, users become active content providers instead of merely being passive readers. Interpretations might help to understand the content of a document. They are also an important means to reconstruct the original context of a document. Annotations may contain reviews and additional information about a document. As we elaborate later, annotations are an important kind of metadata attached to a document. On the collection level, annotations can be employed to link information resources, creating new explicit relationships between them (Neuhold et al., 2004; Agosti et al., 2007a). Furthermore, annotations can support access and retrieval of the information sources managed in a digital library repository – the information contained in annotations may be important to judge the relevance of a document w.r.t. a query, as annotations are a special kind of document context (Frommholz et al., 2004a). How to model this context and employ it for information retrieval is the focus of Part II of this thesis.

Closely related to digital libraries are collaboratories. A *collaboratory*, as formulated by William Wulf, is defined as a

> ...center without walls, in which nation's researchers can perform their research without regard to geographical location – interacting with colleagues, accessing instrumentation, sharing data and computation resource, and accessing information in digital libraries. (Kouzes et al., 1996)

Collaboratories focus on facilitating scientific *interaction* and *collaboration* within a team. Besides this, they should support the *sharing* of data and resources. Annotations can support collaboratories in the above tasks by providing means to share annotations as well as annotation-based discussion for the collaborative interpretation of the given material.

Consequently, the DELOS Network of Excellence on Digital Libraries sees annotation as a new form of communication and identifies the understanding and managing of this new medium as one of the major challenges in DL research (Del Bimbo et al., 2004). Several studies have been performed to assist the design of digital library systems supporting annotations on the user level (Marshall, 1997, 1998), on the conceptual level (Agosti and Ferro, 2003; Agosti et al., 2004, 2007a) as well as on the system level (Agosti et al., 2005a, 2006). Current annotation research also deals with the question of how to anchor annotations to the passage they belong to (Bernheim Brush, 2002).

2.1.2.2 An example: Annotation-based scientific discussion in COLLATE

We present COLLATE as an example of a collaboratory for the humanities which enables scientific discussion through annotations. The COLLATE[3] collaboratory (Thiel et al., 2004) focuses on historic film documentation, dealing with documents about films of the 20s and 30s of the last century. Such documents can be, for example, censorship decisions, newspaper articles, etc. They are digitised and stored in the system repository. COLLATE supports the work between film scientists in different locations by establishing a collaboration cycle (Frommholz et al., 2003): users can react to other users' contribution, and so the cycle continues. Users have the option of manually assigning keywords to the digitised documents as well as cataloguing them according to a pre-defined schema. One of the central concepts of COLLATE is to support document interpretation by enabling scientific discussion about documents through annotation threads comprised of shared annotations.

[3] Collaboratory for Annotation, Indexing and Retrieval of Digitized Historical Archive Material, `http://www.collate.de/`

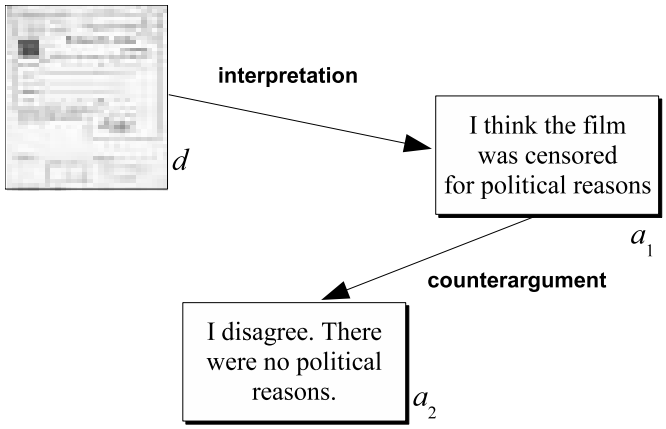


Figure 2.1: An annotation thread in COLLATE

Annotation threads consist of the annotated document (or a part of it) as root and nested annotations connected to the root. The links between the nodes of an annotation thread (documents and textual annotations) are typed with so-called *discourse structure relations*. In COLLATE, the following relations are defined: *elaboration* (giving additional information), *analogy* (describing similarities), *difference* (describing contrasts), *cause* (stating a cause for specific circumstances), *background information* (e.g., information about the background of an author), *interpretation* (of statements), *support argument* and *counterargument* (support or attack other arguments). Figure 2.1 shows an example of two discourse structure relations. The incorporation of these relations is discussed in more detail in Brocks et al. (2002). Modelling annotation threads this way gives us explicit information about the pragmatics of statements (through link types).

Figure 2.2 shows a screenshot of the COLLATE prototype. In the lower right corner we see in the background a page of one of the typical digitised documents film scientists deal with in COLLATE. Users can annotate this page, the whole document or a fragment of the page. To annotate a fragment, the user can mark the fragment with a rectangle, as it happened around the stamp in the example page. Above the digitised page we see a typical annotation. On the left hand side of the screenshot there is a window showing the annotation thread belonging to the document. In front of it, we can see the comment dialogue box for entering a new comment. The user can see the message she is replying to for reference. She can choose from one of the annotation types above and type her message. On the right hand side of the dialogue box the user can request further actions on the new annotation or one the old one, which is one of the various mechanisms in COLLATE to foster collaboration.

2.1.2.3 Other Systems supporting Annotations

There are many other digital libraries and collaboratories which support several kinds of annotations. DAFFODIL (Klas et al., 2004a) is targeted at the support of the *digital library life cycle* proposed by Paepcke (1996). While initially focusing on *strategic retrieval support*, improvements of DAFFODIL concentrate on *interpreting* the material at hand, *sharing* new insights and *creating* new knowledge. To support these tasks, the user is provided with basic annotation functionality like the creation of annotations, browsing of annotation threads and display of particular annotations (Agosti et al., 2006). As an example for the various possible

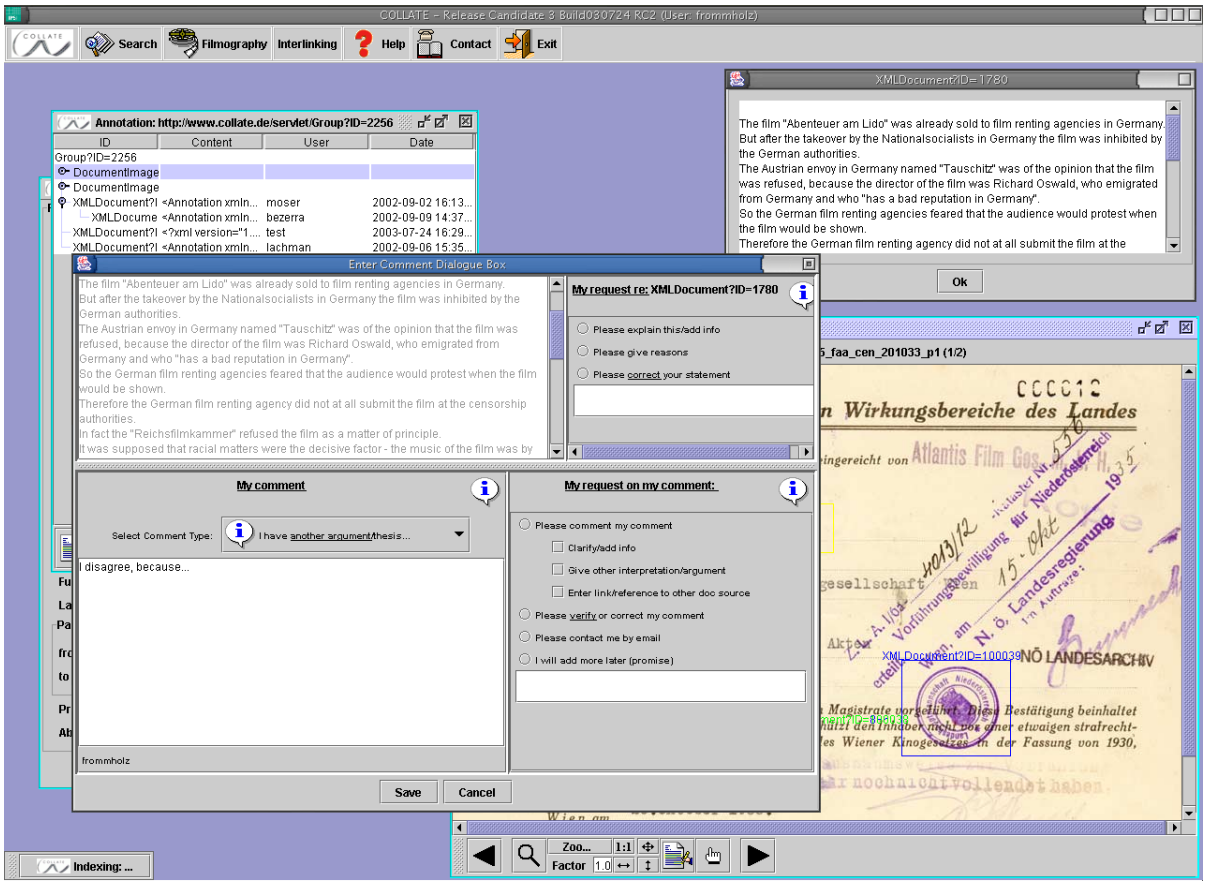


Figure 2.2: Screenshot of the COLLATE prototype

scenarios, users can discuss the content of important documents in their state-of-the-art with collaborators in order to develop new ideas based on previous ones; these ideas can later result in new publications which in turn become part of a DL system's repository. In DAFFODIL, users can furthermore categorise documents relevant for their work in a personal library (PLib) using folders and subfolders. The management of the document content in the PLib is supported by means of private annotations. Annotations and their type in DAFFODIL are modelled as an ontology. In this annotation ontology, each annotation type is an own class inheriting from the generic class Annotation. Annotations themselves are instances of the classes of this ontology. For example, annotations of type "comment" are instances of the Comment class.

Another interesting example of a system supporting annotations is IPSA (Agosti et al., 2005b). Here, linking annotations are used to relate images, for example to make hidden relationships between images and illustrations explicit. To this end, it is possible to state by which image another image was inspired or to see which image is a copy of another one. The typed relations between these images are modelled as annotations using a specific "RelateTo" relationship. The DEBORA project (Nichols et al., 2000), where annotations can be chained together to allow for trails and virtual books consisting of Renaissance documents, goes in a similar direction.

XLibris (Schilit et al., 1998b) is a system supporting active reading through free form digital ink annotations. XLibris runs on a pen tablet PC and supports the paper document metaphor

to capture several important characteristics of paper documents. Users can create margin notes as well as highlight important parts of the text.

From a conceptual and implementation point of view, annotations are often realised as superimposed layers referencing certain points on a base (document) layer (Delcambre et al., 2001; Phelps and Wilensky, 1997).

2.1.3 Annotations on the Web

2.1.3.1 "Web 2.0", Blogs, Forums, Web Annotations

Very similar to annotations in digital libraries and collaboratories are Web annotations. Annotations gain increasing importance in what is called the "Web 2.0" (O'Reilly, 2005) as an instrument to make the Web more interactive and involve the user in creating content. We find annotations in typical flagships of the Web 2.0 paradigm. For example, entries in the free encyclopedia Wikipedia[4] can contain comments discussing the corresponding entry in order to clarify and elaborate things, or discuss possible modifications to an entry. The photo sharing platform Flickr[5] allows users to comment (parts of) pictures and photos. Another typical Web 2.0 concept realised in Flickr (and many other platforms) is *collaborative tagging*, which is "a style of collaborative categorisation of sites using freely chosen keywords, often referred to as tags" (O'Reilly, 2005). In Flickr, for example, users can annotate a photo by assigning it different tags, not bound to any controlled vocabulary or ontology. Another concept which came up in the Web 2.0 context are *blogs*. Blogs (short for "web logs") can be seen as a kind of online diary, since blog entries are chronologically ordered. While blog entries are no annotations per se, they often refer to other Web pages, which make them an annotation of these pages then. Furthermore, many systems provide the possibility to discuss these entries. A variant of blogs is Slashdot.org[6], where comments on interesting Web pages are posted, which in turn can be commented upon. Another typical application for annotations can be found in news portals like ZDNet News[7], where articles about developments in the IT business are posted. Users can comment these articles, and the comments can be annotated again, which enables a discussion which starts with the content of the corresponding article.

Besides Web portals which let users annotate the presented content, there also exist tools to annotate arbitrary Web documents. These systems usually let users select a fragment of a Web page or a whole page and then offer the possibility to write a (private or public) note and/or start a discussion about the content of the page or the selected fragment. Some of these systems also support annotation types. Examples of such systems are Annotea (Kahan et al., 2001), Yawas (Denoue and Vignollet, 2000) and MADCOW (Bottoni et al., 2004). Another application of Web annotation tools is the creation of trails to implement guided tours (Röscheisen et al., 1995; Furuta et al., 1997).

2.1.3.2 Product Reviews

Product reviews are another kind of annotation which can be found on several e-commerce Web portals. For instance, Amazon[8] allows for reviewing books, CDs and other products. Such reviews often contain a scale where users can judge the quality of a product, e.g. by giving

[4] http://www.wikipedia.org/
[5] http://www.flickr.com/
[6] http://www.slashdot.org/
[7] http://news.zdnet.com/
[8] http://www.amazon.com/

0 (very bad quality) to 5 stars (very good quality). Additionally, users can give comments explaining the reasons for their judgement.

2.1.4 Email Discussions and Usenet News

The concept of collaborative annotation is not only observed on the Web or in digital libraries, but also within Internet services which are older than the Web: email discussions and Usenet News. *Email discussion lists* (or *mailing lists*) let users discuss topics in a certain domain using their email client. An example are the discussion lists offered by the World Wide Web consortium[9] (W3C) to discuss topics about the Web and related technologies. A similar technology, although underlying a different protocol and different routing strategies, is Usenet News (Kehoe, 1993, chapter 4), which came up around 1979. Users can subscribe to several newsgroups, which cover a certain domain area and are hierarchically ordered. Email discussion lists and Usenet News have in common that users can start a new discussion thread, triggering other users to write a reply by quoting interesting passages from the previous article; the reply then annotates the selected fragment of the previous email. A further thorough discussion on how newsgroup postings and emails relate to annotations can be found in Section 7.1.

2.1.5 Semantic Annotation

One of the fundamental ideas of the so-called *Semantic Web* is to make data available on the Web machine-readable. The idea is to let human-readable data reside besides machine-understandable data. One of the core building blocks of the Semantic Web is *semantic annotation* (Handschuh and Staab, 2003a). Semantic annotation involves relating the objects of an ontology (a formal, explicit specification of domain concepts and their semantic relations) to documents or document parts. Annotations are then a classification of the selected element as instantiation of an ontology object, an object property or a relation between objects (Handschuh and Staab, 2003b). The assignment of ontology objects to documents can be done manually or semi-automatically (by applying information extraction techniques), employing the resource description framework (RDF)[10]. For example, the author name on the cover of this thesis can be set in relation to a concept "PhD Student". Based on interrelations in the ontology and the semantic annotation, an inferencing system might infer that this thesis was written by a PhD student supervised by Norbert Fuhr.

2.1.6 Scientific Databases

Annotations are also an important issue in scientific databases to support collaborative work and the reuse of its outcome. Annotations in scientific databases let scientists semantically enrich their results or the object under consideration. One example is the Gene Ontology Annotation project (Camon et al., 2003) to let curators annotate entries in the SWISS-PROT protein knowledge base or from other sources. Annotation in this sense is similar to semantic annotation, because a database entry is linked to an object in an ontology. A similar example from bioinformatics is the Distributed Annotation System (DAS) (Dowell et al., 2001), which is concerned with genome annotation. The focus of DAS lies on the distribution of annotation servers. This way, laboratories and their members can contribute to the overall annotation

[9] http://www.w3.org/Mail/
[10] http://www.w3.org/RDF/

effort, but have total control over their annotations since they may have their own annotation server.

A third example comes from neuroscience. Gertz et al. (2002) report about a concept-based approach to annotate scientific images by assigning them to domain-specific concepts. For images, annotations are an important means for semantic indexing, since automatic methods are capable of extracting syntactic features (like colour distribution, shapes, etc.), but usually fail to grasp the meaning of an image. To semantically enrich scientific images, neuroscientists can define a *region of interest* in the image and annotate these regions with concepts of their domain (which, again, is similar to semantic annotation). As within DAS, annotation here is a distributed process as well. This means that inconsistent or incompatible data may be provided for one single region.

2.1.7 Linguistic Annotation

As a last kind of annotation, we briefly present linguistic annotation. *Linguistic annotation* comprises part-of-speech tagging, morphological analysis, chunk analysis and named entity recognition; one of its purposes is to aid information extraction and thus semantic annotation (Buitelaar and Declerck, 2003). A full text is separated into tokens, which can be single words or phrases. Each such token may be annotated with, for example, its syntactic class (e.g., noun, verb, etc.) as a result of part-of-speech tagging, or its named entity (a named entity recognition would, for instance, identify the author name on the cover of this thesis as a person).

2.2 Facets of Annotations

So far we have introduced annotation systems, concepts and usage. In the following, we describe the different facets of annotations that we consider by repeating and elaborating the findings in Agosti et al. (2004).

2.2.1 Annotations as Metadata

Metadata is defined as "data about data". Examples of metadata are bibliographic or descriptive metadata; DublinCore metadata records[11], for instance, contain fields where, e.g., the creator and title of a document, as well as a description of the document content and a rights management statement can be provided (Arms, 2001, chapter 10).

From a syntactic point of view one of the main characteristics of metadata is that it is connected to the object it refers to. Annotations have a similar connection to what they are annotating, and they are considered as additional data about an existing data, that is annotations are metadata (Nagao, 2003). This reflects a *data specific* view on annotations. For example, the World Wide Web Consortium (W3C) considers annotations as metadata and interprets them as the first step in creating an infrastructure that will handle and associate metadata with content towards the Semantic Web. Kahan et al. (2001) state that "from a general viewpoint, annotations can be considered as metadata; they associate remarks to existing documents". This viewpoint also implies that not only structured annotations (like we have with semantic annotation) can be considered metadata, but also semi-structured or unstructured comments.

[11] `http://dublincore.org/`

2.2.2 Annotations as Content

Another view on annotations is seeing them as *content*, reflecting an *information specific view*. Annotations can be regarded as content in two ways: they can be *content about content* and they can be considered as *additional content* (Nagao, 2003). Annotations being additional content extend the content in the annotated document, for instance by means of elaboration (e.g., "it is Paris in Texas, not in France"), examples, arguments and counterarguments and interpretation. We say that an annotation is created on the *content level* in this case. Annotations on the content level also have another special property, which Agosti et al. (2007a) and Agosti and Ferro (2003) describe as the dualism between annotations as content enrichment and as stand-alone content. In the former view, annotations are not autonomous, but rely on the annotated object as information resource to justify their existence. In the latter view, they are considered as real documents and autonomous entities. When we first start annotating a document, this annotation, as an outcome of active reading, is very closely related to what we previously read and can be seen as an extension of it. But on the other hand, the annotations can contain new insights, interpretations or ideas, which are new content in its own right. This way, annotations bridge the gap between reading and writing (Marshall, 1998).

In contrast to the view of annotations on the content level, annotations can be regarded as content on the *meta level*. By this we mean annotations which do not extend existing content, but its content says something *about* the content of the annotated object or the annotated object itself. Examples are annotations being reviews and judgements (e.g., saying "this is a good and well-written introduction" or "this book is certainly too expensive"); they make assertion about the content of the annotated object, for example if they judge the quality of an argument ("there is no reason to assume this"), or even contain interpersonal statements about the author of an annotated object ("the author is well-recognised in her community" or "he should do his homework first"), but do not necessarily extend the content. We call such annotations *meta level annotations* (or short *meta annotations*), which are not to be mixed up with the concept of metadata, which simply is another facet of annotations and reflects a syntactic viewpoint – both content and meta level annotations are metadata from a syntactic point of view. The distinction between content and meta level annotations is purely made with respect to a semantic viewpoint. We can find other examples of meta level annotations in Wolfe (2000). Note that both ways do not necessarily mutually exclude each other: interpretations, for example, may be content about content, but they might also contain additional content.

2.2.3 Annotations as Dialogue Acts

Another viewpoint on annotations is regarding them as *dialogue acts*. This covers a *communication specific view*, which is concerned with the question of the *pragmatics* conveyed in annotations, i.e. the intention behind a user's statement. Gaining information about pragmatics is an important means to distinguish between the different kinds of content we have discussed when regarding annotation as content. We may find out about the semantics of utterances in annotations, but this does not necessarily mean that we can distinguish whether we can see the annotation as content about content or an extension of existing content.

Each annotation implicitly consists of certain *communicative acts*, which, according to Searle, can be classified as (among others) *assertives*, *directives* (e.g., requests), and *commissives* (e.g., promises) (Searle, 1979). Communicative acts both allow for communication on the content and on the meta level. On the content level, assertives connected with a certain *discourse structure relation* are the units with which a coherent interpretation of the material can be

created (Brocks et al., 2002). On the other hand, directives and commissives can trigger further collaborative acts on the meta level. Directives can be used to attempt to get some other person to do something; an example would be when a user asks the author of a comment if he could further elaborate on it. The author, in turn, can answer the request with a promise to provide the needed information (and actually provide it later on). Certain communicative acts can thus enable collaboration, and they can be realised as annotations.

2.2.4 Annotations as References

Annotations may be references which link together two different documents or parts of documents. A classical example for such a reference annotation is an arrow drawn from one text passage to another on the same page in order to bring them in relation. It could also be a text like "see also the paper I read last week". So a reference annotation can be combined with a textual sign (Agosti and Ferro, 2003), meaning that we not only have a link between two objects, but also some text describing the type or even the semantics of the link. This way, documents and objects in the whole repository can be brought in relation. With reference annotations, resources can be chained into paths like it is done in the Walden's Path system (Furuta et al., 1997).

2.2.5 Polarity of Annotations

Another facet of annotations is their *polarity*. Annotations might convey a positive or negative sentiment towards the object they are annotating. As an example on the content level, a counter argument relation type, as it is found in COLLATE, indicates a negative polarity towards an argument in the annotated object, whereas a support argument would implicitly convey a positive sentiment. On the meta level, reviews of and judgements about an object might overall be positive or negative.

2.2.6 Annotations and Hypertexts

When annotations are references, they link together the objects contained in the repository, thus establishing a huge hypertext, according to the definition of hypertext provided in Agosti and Smeaton (1996). Marshall (1998) considers annotations as a natural way of enhancing hypertexts by actively engaging users with existing content in a digital library (Marshall, 1997). But also annotations not being references (like textual comments) are part of this hypertext, due to their strong relation to the annotated object. Tools like the Multivalent Browser (Phelps and Wilensky, 1997) let users select a text fragment which is then to be annotated, and also in COLLATE it is possible to choose a part of an image as a fragment for annotation. So we might regard such an annotated fragment (or document part or passage) as a single node in this hypertext. We gain a web consisting of objects (documents or annotations) and their fragments which are connected through different kind of links. For instance, annotated fragments can be related to their original object they are contained in with an *is fragment of* relation; nodes in an annotation thread are connected via an *annotates* or *has annotation target* relation, while annotations being references introduce a specific *references* relation. We call the resulting structure the *annotation hypertext* (elsewhere also referred to as *document-annotation hypertext*). If we do not regard documents and annotations as atomic units, but also consider their internal structure (like books made of chapters, chapters made of sections), each of these structural elements can be a node in this hypertext in its own right, connected to its related structural elements through an *is part of* link. We call such an extended hypertext which combines intra-

and inter-document relations as well as annotations and annotated fragments the *structured annotation hypertext*. Structured annotation hypertexts and their properties are going to be discussed thoroughly in the next chapter.

2.3 Summary and Discussion

In this chapter we have examined digital annotations, their usage, properties and facets. Annotations play an important part in digital libraries and collaboratories as well as on the Web, as COLLATE and many other applications show. They are the building blocks for scientific discussion in order to collaboratively interpret the material at hand. We can even regard email discussions and newsgroup postings as annotations in a broader sense. In another manifestation, annotations are a key concept when talking about the "Web 2.0", the Semantic Web or in scientific databases. They can furthermore link repository resources when they are references. Their main usages are to remember, think, clarify and share, but they can also be a starting point for creating new knowledge or even contain new information. From a syntactic point of view, annotations are metadata. From a semantic viewpoint, they are either content about content (in case of meta level annotations) or additional content (in case of content level annotation). We can view them as objects which are strongly connected to the annotated entity, but also as stand-alone objects which are equally important as the main documents in a repository. Another facet of annotations, from a pragmatic point of view, is to see them as dialogue acts. Additionally, annotations can convey a positive or negative sentiment towards the entity they belong to. As annotations are connecting the various objects found in a repository and in turn are connected to the objects they annotate, they establish an annotation hypertext. In case the objects in a repository are structured and we make the document structure explicit, the annotation hypertext can be expanded to form a structured annotation hypertext.

Which early conclusions can we draw for information retrieval? From the considerations above it is clear that on the one hand, we need retrieval mechanisms which let users search for annotations, since otherwise important insights and new ideas might never be uncovered (especially if we see annotations as stand-alone documents). On the other hand it also becomes clear that annotations are an important additional source to decide the relevance of documents which are annotated, so retrieval functions should consider the annotation context. We have identified important properties of annotations which should be considered when creating annotation-based retrieval methods. First of all there is the fact that annotations, annotated fragments and documents establish a hypertext. This means that all nodes in this hypertext are embedded in a context given by the links to other objects. Annotation-based retrieval methods should take this context under consideration. Second, many systems offer the possibility to type annotations. Annotations of various type might play a different role in the retrieval process, especially when it comes to distinguishing between content level and meta level annotations. Third, a very interesting side-effect of annotation is the identification of the annotated fragment or passage, which seems to be an important part to the annotator as she took the time to highlight this part or even write a comment about it. So fragments are another player when it comes to annotation-based retrieval. And last but not least, also the polarity of annotations might have an influence on the decision whether an object is relevant or not.

The findings in this chapter help us to understand the concept of annotation and are an important basis for the design of annotation-based retrieval functions. The next step is to

formalise the insights we gained in order to make them ready to be exploited by novel retrieval functions. This is the main issue of the following chapter, which deals with the definition of the structured annotation hypertext and a discussion of its properties.

3

A Model of the Annotation Universe for Annotation-based IR

That's why we like you, Mulder.
Your ideas are weirder than ours.

(Lone Gunman to Fox Mulder, "The X-Files")

If we talk about annotation-based retrieval, we first have to determine the data structure our methods can operate on. What are the objects, relations and constraints we are potentially dealing with when we talk about annotation-based retrieval? In the last chapter we learnt about certain features of annotations and we also mentioned structured annotation hypertexts. Based on these considerations, the aim of the model discussed in this chapter is to make explicit the objects and relationships which play an important part in annotation-based retrieval, either as being a retrievable object (i.e. an object which can be retrieved by the system) or as providing the context in order to determine the relevance of objects. So the model here is tailored to the task of annotation-based information retrieval and does not claim to be thorough w.r.t. every possible property of annotations and annotated objects. Our model reflects, whenever possible, previous formal models for digital libraries and annotations of digital content, as they are introduced in the 5S model by Gonçalves et al. (2004), and by Agosti and Ferro (2007), respectively. It further emphasises and integrates structured documents, as they are used for structured document retrieval (Fuhr et al., 2002). The model is simple; it should not be regarded as yet another model of annotations in digital libraries – readers interested in that are referred to the aforementioned publications.

We will first discuss an object-oriented view on annotations which identifies important objects and relations. One the instance level, our discussion leads us to our core data structure, the annotation hypertext, and an extension of it, the structured annotation hypertext. We use Description Logics (Baader et al., 2003) to express the annotation model. With Description Logics, it is possible to exactly specify objects and their properties and relationships, especially inheritance relations[1]. Basically, there are two key components to model a domain in Description Logics: The *TBox* and the *ABox*. While the TBox is used to specify the concepts and terms of our domain, together with their properties and relationships, the ABox is used to

[1]Note that Description Logics are only used as the language to specify the data structure. In the next chapters, another logical framework will be introduced which implements the actual retrieval approach based on the data structure defined here.

specify the individuals in our universe. In the next section, we are going to discuss the TBox containing the classes we deem important for our aims. In the then following section, the ABox is used to extract and discuss the basic data structure, the structured annotation hypertext. Instead of giving an overview of the Description Logics syntax, Description Logics constructs are briefly introduced when they are needed. The interested reader is referred to (Baader et al., 2003, chapter 1) for an introduction to Description Logics and their semantics.

3.1 Main Classes

In this section we define the terminology of our annotation universe by defining the classes and properties of our TBox or object-oriented view, respectively. The objects and main relations of view are also depicted in Figure 3.1, which summarises the further discussion. An early version of the model was discussed before in Frommholz and Fuhr (2006b).

3.1.1 Digital Objects

A digital library manages digital objects of various kinds (Gonçalves et al., 2004). Such digital objects might be, e.g., textual and multimedia documents, but also other kinds like digital representations of persons and conferences, as we find them for example in the DAFFODIL system (Klas et al., 2004b). In principle, annotation-based retrieval approaches might return any digital object in the repository when processing a query. Those digital objects need to be

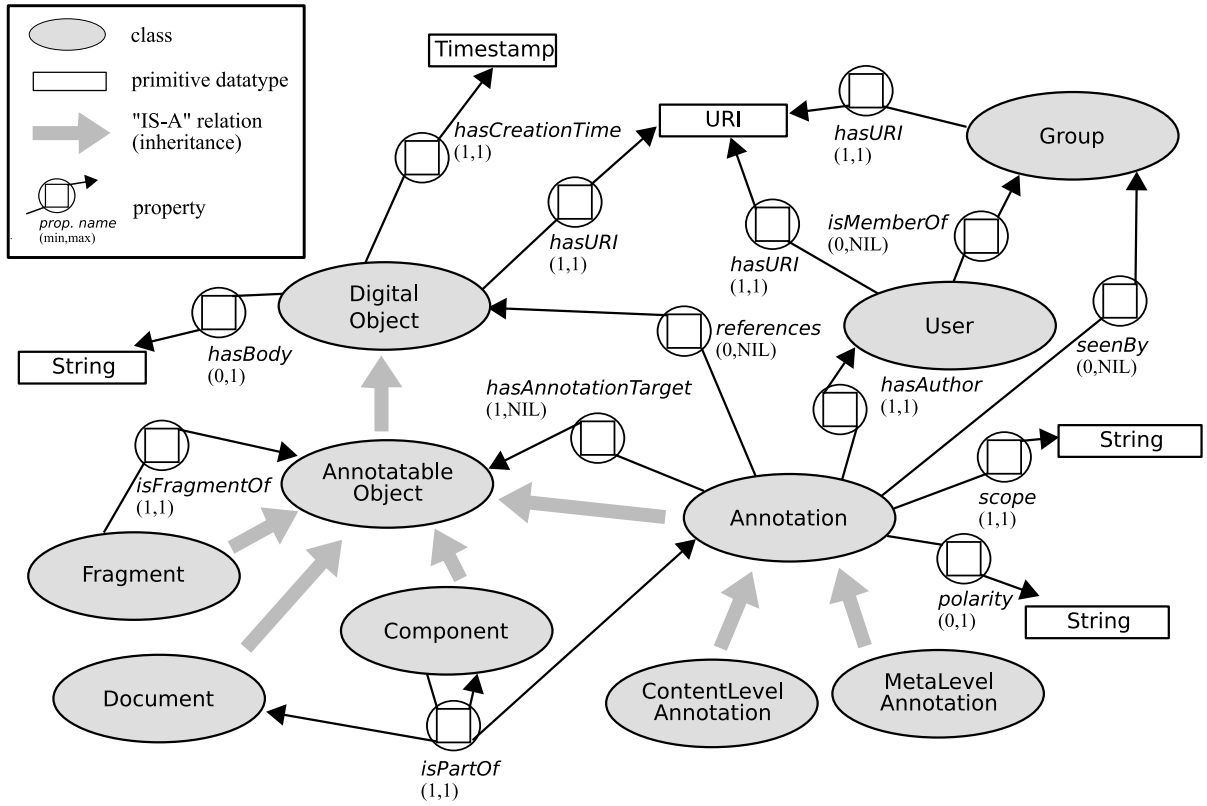


Figure 3.1: Classes and properties of the object-oriented view (the TBox)

uniquely identified. In our case, we say that every digital object has a Uniform Resource Identifier (URI) (Berners-Lee et al., 2005); of course, other identification mechanisms are possible and subject to the actual application[2]. We also record the creation time of any digital object.

We will now identify the first class, DigitalObject, as an object which is identified by a URI. In Description Logics, the definition of a class actually means the definition of a subset of individuals with certain properties. When specifying a class C, we mean all individuals which can be classified as C. So DigitalObject denotes all individuals which can be classified as digital objects. $A \sqcap B$ describes all individuals which are in both A and B. We say that all individuals identified by a URI are digital objects, and all digital objects must be specified by a URI, which we express with a property called hasURI. The set of all individuals having exactly 1 hasURI relation with a `URI`[3] primitive datatype is expressed in Description Logics as

$$(= 1\ \mathsf{hasURI}) \sqcap \forall \mathsf{hasURI}.\texttt{URI}. \tag{3.1}$$

$(= 1\ \mathsf{hasURI})$ specifies the set of all individuals having exactly 1 hasURI property. Generally in Description Logics, $\forall P.C$ is the set of individuals that are in the relationship P with individuals belonging to the set denoted by the concept C. So $\forall \mathsf{hasURI}.\texttt{URI}$ determines the set of individuals for which the range of the property hasURI is `URI`, and $(= 1\ \mathsf{hasURI}) \sqcap \forall \mathsf{hasURI}.\texttt{URI}$ are all individuals which have exactly one URI and whose URI is of type `URI`. Similar to that,

$$(= 1\ \mathsf{hasCreationTime}) \sqcap \forall \mathsf{hasCreationTime}.\texttt{Timestamp}. \tag{3.2}$$

specifies the set of all individuals having exactly one unique time stamp as creation time.

A digital object may have a body which contains its content. For the sake of simplicity, we see the body as a string and therefore use the primitive datatype `String`. However, other datatypes are conceivable, even complex ones. For example, `xsd:base64Binary` (Biron and Malhotra, 2004) may be used to encode binary content like multimedia data in a string. Gonçalves et al. (2004) introduce the notion of *streams*, which is a more general one and may be applied here as well. The connection between digital objects and their bodies is given by the hasBody property, so we define the set of individuals having at most 1 body:

$$(\leq 1\ \mathsf{hasBody}) \sqcap \forall \mathsf{hasBody}.\texttt{String} \tag{3.3}$$

where $(\leq 1\ \mathsf{hasBody})$ is the set of individuals having 0 or 1 property hasBody.

(3.1) to (3.3) describe three sets of individuals with certain properties. We define the class DigitalObject as the set of individuals which belong to all these three sets:

$$\begin{aligned} \mathsf{DigitalObject} \equiv\ & (= 1\ \mathsf{hasURI}) \sqcap \forall \mathsf{hasURI}.\texttt{URI} \\ \sqcap\ & (\leq 1\ \mathsf{hasCreationTime}) \sqcap \forall \mathsf{hasCreationTime}.\texttt{Timestamp} \\ \sqcap\ & (\leq 1\ \mathsf{hasBody}) \sqcap \forall \mathsf{hasBody}.\texttt{String} \end{aligned}$$

In Description Logics, the expression $A \equiv B$ says that A and B are equivalent – any individual which is in A must also be in B and vice versa.

[2] Taking URIs is one possible option. It is of course possible to use the more general notion of *unique handles* for digital objects as it is done in Gonçalves et al. (2004). These handles may be URIs, but also, e.g., Digital Object Identifiers (DOIs). Since URIs are meant to not only identify digital objects, but, for instance, persons as well, we use URIs where we could have used handles instead. Instead of URIs, any other identification scheme is conceivable here.

[3] From a Description Logics point of view, primitive datatypes are classes as well. But in order to distinguish them from the other classes in our model, we write them with typewriter fonts.

3.1.2 Structured Documents

Although any conceivable object may be a digital object, the most important ones we are dealing with are documents. We use the term "document" for any main item in a digital library which conveys some kind of content. Documents may thus be textual documents, but also multimedia ones, for instance. In our model, documents are digital objects and we do not introduce new properties for them:

$$\mathsf{Document} \sqsubseteq \mathsf{DigitalObject}.$$

In Description Logics, $A \sqsubseteq B$ means that every individual in A is also an individual in B; the expression $A \sqsubset B$ further specifies that there is at least one individual in B which is not in A. '$\sqsubseteq$' thus models the *IS-A* relation with inheritance between documents and digital objects: each document is also a digital object. Particularly, each document inherits the properties from digital objects: it has exactly one URI and it possibly has a body.

If the structure of a document is made explicit, we talk of *structured documents*. In a document-centric view, we can think of books containing parts, parts containing chapters, chapters containing sections and so on. In a data-centric view, a "document" is a kind of formatted data like spreadsheets and database records which have a certain structure (Fuhr et al., 2002). In both cases, we can identify subcomponents which are part of exactly one supercomponent. For this, we introduce the isPartOf property. This property thus models the structure of a document as an aggregation of subcomponents. Note that each subcomponent might again be structured. So a component may be part of a document or of another component (which in turn must be part of another supercomponent). We therefore define the Component class as

$$\mathsf{Component} \equiv (= 1\ \mathsf{isPartOf}) \sqcap \forall \mathsf{isPartOf}.(\mathsf{Document} \sqcup \mathsf{Component}). \tag{3.4}$$

Components are part of exactly 1 other object, and this object must be a document or another component. Furthermore, we determine that components are digital objects as well:

$$\mathsf{Component} \sqsubset \mathsf{DigitalObject}. \tag{3.5}$$

Note that we used '$\sqsubset$' instead of '$\sqsubseteq$' – the case that $\mathsf{Component} \equiv \mathsf{DigitalObject}$ is impossible since there has to exist at least one digital object which is not a component. Again, the properties of digital libraries are inherited by components, so they might have a body and we specify that they are identified by a URI.

Modelling components as digital objects reflects a different view than expressed in Gonçalves et al. (2004). Here, each digital object might contain a set of *structured streams* in case the digital object is a structured document (like, e.g., an XML document). Such a structured stream consists of a *structure*, which is a directed graph, and a *stream* (see above). Each node of the structure is associated with a segment of the stream, but the node is not seen as a digital object in its own right. This is different in our model, where we want to retrieve subcomponents as well, and therefore need to address them. For the sake of simplification, we also see components as digital objects in our model.

We further assume that the graph representing a structured document forms a tree (this is consistent with the view of XML document trees representing the structure of a document (Fuhr and Großjohann, 2004)). Gonçalves et al. (2004) define an entailment relation for subparts of a structured document, saying that a stream segment of a node contains the segment of each subnode. In our model, we would form the body of a component with the part of the stream which belongs to the component's node and only this node. For example, consider a stream

$s =< s_1, \ldots, s_n >$ and a component c_1 with subcomponents c_{11} and c_{12}. Let us further assume that s is the stream belonging to c_1. Furthermore, there are two substreams $s1 =< s_i, \ldots, s_j >$ and $s1 =< s_k, \ldots, s_l >$ with $1 \leq i \leq j < k \leq l \leq n$, and $s1$ associated to c_{11} and $s2$ associated to c_{12}. In that case, the body of c_{11} would be made of $s1$ and the body of c_{12} would be made of $s2$. The body of c_1 would contain $s \setminus (s1 \cup s2)$. So the body of a subpart is not contained in the body of its superpart. The reason is that our retrieval approach presented in the second part of this thesis applies a technique called knowledge augmentation to aggregate these parts again in an appropriate way.

3.1.3 Annotatable Objects and Annotations

After having defined digital objects and structured documents, we now come to the core elements of our model – annotations and the objects they annotate, which we call annotatable objects. As outlined in Chapter 2, annotations can also be references connecting several digital objects and, like documents, they can also be structured. From a retrieval perspective, it is also interesting to ask for the author of an annotation and the groups she belongs to.

3.1.3.1 Annotatable Objects

An *annotatable object* is a digital object which can be annotated (in contrast to those digital objects which might not be annotatable). We define the new class of annotatable objects as a subclass of digital objects:

$$\mathsf{AnnotatableObject} \quad \sqsubseteq \quad \mathsf{DigitalObject}.$$

Each annotatable object therefore inherits the properties from DigitalObject. We further say that structured documents and their components can potentially be annotatable, so we have

$$\begin{aligned} \mathsf{Document} &\quad \sqsubseteq \quad \mathsf{AnnotatableObject} \\ \mathsf{Component} &\quad \sqsubseteq \quad \mathsf{AnnotatableObject} \end{aligned}$$

3.1.3.2 Annotations

So far we discussed annotatable objects without introducing annotations themselves. Annotations are objects as well, so we introduce a class

$$\mathsf{Annotation} \sqsubset \mathsf{DigitalObject}.$$

As explained in Section 2.2.1, annotations are metadata, strongly connected to the object they refer to. This means that annotations cannot exist as independent entities from a syntactic viewpoint, but must have at least one specific *annotation target*, which can be any annotatable object in the digital library repository. This is expressed by the property hasAnnotationTarget. So we say that each individual which has at least one annotation target is an annotation, and annotations must have at least one annotation target which is an annotatable object. In Description Logics, we can express this as

$$\begin{aligned} \mathsf{Annotation} \quad \equiv \quad & (\geq 1\ \mathsf{hasAnnotationTarget}) \sqcap \\ & \forall \mathsf{hasAnnotationTarget}.\mathsf{AnnotatableObject}. \end{aligned}$$

which also means that, in contrast to the model in Agosti and Ferro (2007), an annotation can have more than one target it belongs to. We allow for multiple annotation targets for three

reasons: first, the model in Agosti and Ferro (2007) does not forbid the annotation of one or more parts in a digital object (only the annotation of more than one digital object). Such parts might be components as introduced above, and we see each of them as a digital object in its own right; annotating more than one of them would then violate the constraint that only one digital object can be annotated. So if a system allows for annotating more than one part in a document, and we assign to each part an identifier and regard it as a retrievable object in our model, we must weaken the constraint reported in Agosti and Ferro (2007) and permit annotations to have more than one annotation target. The second reason is that we should not exclude possible systems which allow for the annotation of multiple digital objects, for instance to enable the annotation of sets of components or groups of digital objects. Third, as we will see in Section 7.1.2, emails could contain annotations which belong to fragments from different other emails, which means they annotate more than one object. If we want to consider this annotation view on emails as well in our model, we must allow multiple annotation targets.

Another important feature is to nest annotations, i.e. make them annotatable again, which is crucial when modelling annotation-based discussions. So

$$\mathsf{Annotation} \sqsubset \mathsf{AnnotatableObject}.$$

3.1.3.3 Annotations as References

As discussed in the last chapter, annotations might be links or references connecting an annotatable object with other digital objects (in contrast to hyperlinks given by the creator of a digital object). We introduce a new property references to establish links between digital objects: each annotation target is a source of the link, whereas each digital object, regardless of being annotatable or not, may be its destination. Annotations may have no or multiple references.

$$\mathsf{Annotation} \sqsubseteq (\geq 0\ \mathsf{references}) \sqcap \forall \mathsf{references}.\mathsf{DigitalObject}.$$

In principle, also documents can contain references (the aforementioned hyperlinks given by the creator of a document), so

$$\mathsf{Document} \sqsubseteq (\geq 0\ \mathsf{references}) \sqcap \forall \mathsf{references}.\mathsf{DigitalObject}.$$

In this work, we concentrate on references established by annotations, so document references only play a minor role.

3.1.3.4 Structured Annotations

Annotations can be structured as well; in this case we talk of *structured annotations*. For example, the MADCOW system introduces structured annotations (Bottoni et al., 2004). To model this, we have to extend our definition of components in (3.4) to

$$\mathsf{Component} \equiv (= 1\ \mathsf{isPartOf}) \sqcap \forall \mathsf{isPartOf}.(\mathsf{Document} \sqcup \mathsf{Component} \sqcup \mathsf{Annotation}) \tag{3.6}$$

so that components can also be part of annotations. Note that components of annotations are not annotations themselves; the hasAnnotationTarget property is not defined for them, but only for the annotation they are a subcomponent of.

3.1.3.5 Users and Groups

We also model that annotations have exactly one author, who in our model is a user of the system and belongs to the (not further specified) class User:

$$\mathsf{Annotation} \sqsubseteq (= 1\ \mathsf{hasAuthor}) \sqcap \forall \mathsf{hasAuthor}.\mathsf{User}.$$

Note that the specification of the hasAuthor property for annotations does not mean that digital objects in general cannot have any author at all; we distinguish here between authors of annotations, which are users, and authors of documents, which often are external persons.

Although we do not want to further discuss the properties of users, there is one important feature which is interesting for annotation-based retrieval: users might belong to one or more group, represented by the class Group and the isMemberOf property. We can thus partially define User as

$$\mathsf{User} \sqsubseteq (\geq 0\ \mathsf{isMemberOf}) \sqcap \forall \mathsf{isMemberOf}.\mathsf{Group}.$$

In the tradition of the Web, users and groups can also be identified by URIs:

$$\begin{aligned} \mathsf{User} &\sqsubseteq (= 1\ \mathsf{hasURI}) \sqcap \forall \mathsf{hasURI}.\mathtt{URI} \\ \mathsf{Group} &\sqsubseteq (= 1\ \mathsf{hasURI}) \sqcap \forall \mathsf{hasURI}.\mathtt{URI} \end{aligned}$$

We do not discuss the classes User and Group and their properties any further. We just remark that users and groups play an important part when talking about the scope of an annotation later, which can be private (i.e., belonging to a certain user), shared (belonging to a certain group) or public.

The model so far still has some shortcomings: neither is it possible to annotate arbitrary parts of documents (document fragments), nor can we define annotation types. These enhancements of the basic model will be introduced in the next two subsections.

3.1.4 Fragments

Fragments are certain portions of digital objects, in the case of an annotation scenario these are areas selected by the user as an annotation target. Such a fragment can be, for instance, a passage or paragraph of a text document, a certain video sequence or an excerpt of an image. When applying the 5S model, fragments might be segments of a stream. The ability to identify a fragment of a document and to annotate it is a crucial functionality in a digital library system supporting annotations. To reflect this in our model, we introduce a new class

$$\mathsf{Fragment} \sqsubseteq \mathsf{AnnotatableObject}.$$

Since fragments are created during the annotation process, they are related to the annotatable object they are part of. To connect fragments and their source object, we introduce a new property isFragmentOf and say that a fragment must belong to exactly one annotatable object:

$$\mathsf{Fragment} \equiv (= 1\ \mathsf{isFragmentOf}) \sqcap \forall \mathsf{isFragmentOf}.\mathsf{AnnotatableObject}.$$

3.1.5 Annotation Types

Some annotation systems offer a categorisation of annotations into several types. Furthermore, according to the considerations in Section 2.2, annotations can contain additional content or content on the meta level, and thus be categorised into *meta level* and *content level annotations*. Explanations, for example, contain additional content and expand the content of the object they refer to. On the other hand, highlight markings operate on the meta level; if a passage is highlighted, the implicit assertion is "this part is important", but there is no additional content. Another example of meta level content are judgements, where people state their opinion about a document. To distinguish between these kinds of annotation types, we create new classes

$$\mathsf{ContentLevelAnnotation} \sqsubseteq \mathsf{Annotation}$$

and

$$\mathsf{MetaLevelAnnotation} \sqsubseteq \mathsf{Annotation}$$

and categorise our example annotation types accordingly, for example:

$$\begin{aligned} \mathsf{Highlighting} &\sqsubseteq \mathsf{MetaLevelAnnotation} \\ \mathsf{Judgement} &\sqsubseteq \mathsf{MetaLevelAnnotation} \\ \mathsf{Explanation} &\sqsubseteq \mathsf{ContentLevelAnnotation} \end{aligned}$$

3.1.6 Scope, Permission and Polarity

Annotations can be private, shared or public. In the first case, only the creator of an annotation has the right to access it; in the second case, the annotation is visible to a whole group. In the last case, everyone can see the annotation. We thus need to model the *scope* of an annotation, expressed by the scope property. It reflects whether the annotation is public, private or shared, so possible values of this property should be like that. In order to ensure authorised access to annotations, the scope goes together with permissions. Shared annotations might be seen by several groups, and not necessarily only the group the author is member of. Annotation-based retrieval functions should actually return only those annotations which are accessible by the current user, which can either be because the annotation is public, or it is shared and the user belongs to a group the annotation is to be seen by, or it is private and the user is the author of the annotation[4]. In any case, we need to know by which groups an annotation can be seen in order to properly handle access to shared annotations. We therefore introduce the property seenBy between Annotation and Group. An annotation might be seen by one or more groups or no group at all (in case of private annotations). Public annotations imply that they can be seen by any group. We do not further discuss mechanisms to prohibit scope and permission conflict as they are not in the focus of this thesis; readers interested in these issues are referred to Agosti and Ferro (2007).

Another attribute of annotations is their *polarity* (if known). For example, annotation types like "agreement" and "support argument" have a clear positive polarity, since they express a positive sentiment about the annotated part. In contrast, "disagreement" and "counterargument" convey a negative sentiment about the annotated content. In these cases, the annotation type determines the overall polarity of the annotation. In other cases, the polarity might not be clearly derivable from the annotation type, so the polarity might be determined by the annotation content (for example, a comment itself does not have a certain polarity, but there

[4] An appropriate access policy is due to the actual application and shall not be discussed here.

can of course be negative or positive statements in the comment). For Annotation, we define a new functional property polarity which might take the values "positive" or "negative".

Scope and polarity are modelled in Description Logics as

$$\begin{aligned} \mathsf{Annotation} &\sqsubseteq (\leq 1\ \mathsf{polarity}) \sqcap \forall \mathsf{polarity}.\texttt{String} \\ \mathsf{Annotation} &\sqsubseteq (= 1\ \mathsf{scope}) \sqcap \forall \mathsf{scope}.\texttt{String}. \\ \mathsf{Annotation} &\sqsubseteq (\geq 0\ \mathsf{seenBy}) \sqcap \forall \mathsf{seenBy}.\mathsf{Group}. \end{aligned}$$

3.1.7 Multiclassification

So far we specified the properties and class assignments of individuals. But what happens if an individual has, say, both the isFragmentOf and hasAnnotationTarget property? In this case, it would be both a fragment and an annotation, which should not be possible. While such multiclassification is obviously desirable for IS-A relations, we have to prohibit this in other cases.

User individuals must not be Group or DigitalObject individuals the same time; similarly, Group individuals must not be User or DigitalObject individuals the same time

$$\begin{aligned} \mathsf{User} &\sqsubseteq \neg\mathsf{Group} \sqcap \neg\mathsf{DigitalObject} \\ \mathsf{Group} &\sqsubseteq \neg\mathsf{User} \sqcap \neg\mathsf{DigitalObject} \end{aligned}$$

Furthermore, we have to ensure that an annotatable object only belongs to one of the classes Fragment, Document, Component or Annotations (or its subclasses). First of all, we say that documents must neither have the isFragmentOf nor the isPartOf nor the hasAnnotationTarget property:

$$\begin{aligned} \mathsf{Document} \sqsubseteq\ & (= 0\ \mathsf{isPartOf}) \sqcap \\ & (= 0\ \mathsf{isFragmentOf}) \sqcap \\ & (= 0\ \mathsf{hasAnnotationTarget}). \end{aligned}$$

This tells us that no document can also be an annotation, a fragment or a component. Note that due to the fact that documents (like annotations, components and fragments) are digital objects, an individual which is classified as a document cannot be a group or a user, too.

We have to define similar restrictions for fragments, components and annotations:

$$\begin{aligned} \mathsf{Fragment} &\sqsubseteq (= 0\ \mathsf{isPartOf}) \sqcap (= 0\ \mathsf{hasAnnotationTarget}) \\ \mathsf{Component} &\sqsubseteq (= 0\ \mathsf{isFragmentOf}) \sqcap (= 0\ \mathsf{hasAnnotationTarget}) \\ \mathsf{Annotation} &\sqsubseteq (= 0\ \mathsf{isPartOf}) \sqcap (= 0\ \mathsf{isFragmentOf}) \end{aligned}$$

We have now modelled our annotation universe. The whole model is shown in Appendix A.

3.2 Structured Annotation Hypertext

In the last subsection we described the classes contained in our logical view. Actual documents and annotations are instances or individuals of DigitalObjects or its subclasses, respectively. On the instance level (or ABox, to use a term from Description Logics), we can derive an important data structure: structured annotation hypertexts. We describe the definition of a structured

annotation hypertext (or structured document-annotation hypertext), which is similar to the one in Agosti and Ferro (2005), but with certain differences, as it deals with fragments and structured documents. For our further considerations, $C(o)$ means that an instance o belongs to a class C, while $R(a, b)$ means instance a has value b for the property R. Before we continue, we need to prohibit that, for example, an annotation annotates or references itself or a fragment is a fragment of itself, or that a subcomponent is part of itself:

Constraint 1 (Loopless): The property value of an instance cannot be the instance itself (but there may be properties having (other) instances of the same type):

$$R(a, b) \Rightarrow a \neq b$$

for each property R. □

We further must prohibit that annotations reference the same objects they annotate (Agosti and Ferro, 2007):

Constraint 2 (Annotation targets and references): If an annotation a annotates an object o and references an object o', they must not be the same:

$$\mathsf{hasAnnotationTarget}(a, o) \wedge \mathsf{references}(a, o') \Rightarrow o \neq o'$$

for all instances a, o, o'. □

We begin our further considerations with the definition of the annotation hypertext and a discussion of some important properties. The annotation hypertext can be derived from the instances of our model.

3.2.1 Annotation Hypertext

An annotation hypertext is composed of digital objects as nodes and specific relations among them as edges.

Definition 1 (Annotation Hypertext):
An *annotation hypertext* (or document-annotation hypertext) is a labelled digraph $H = (V, E)$ with N as the set of vertices and $o \in N$ iff $\mathsf{DigitalObject}(o)$. $E \subseteq V \times V$ is the set of edges and $l : E \longrightarrow \Sigma^*$ is a labelling function over an alphabet Σ. The annotation hypertext is derived from the instances of our model as follows:

- $(n, m) \in E$ if $\mathsf{hasAnnotationTarget}(n, m)$;
- $(n, m) \in E$ if $\mathsf{references}(n, m)$;
- $(n, m) \in E$ if $\mathsf{isFragmentOf}(n, m)$.

Other properties are not considered in the annotation hypertext. For each $e = (n, m) \in E$ it is

$$l(e) = \begin{cases} \text{“hasAnnotationTarget”} & \text{iff} \quad \mathsf{hasAnnotationTarget}(n, m) \\ \text{“references”} & \text{iff} \quad \mathsf{references}(n, m) \\ \text{“isFragmentOf”} & \text{iff} \quad \mathsf{isFragmentOf}(n, m) \end{cases}$$

We can identify a *temporal dimension* of annotations, which regulates the temporal order among annotations and annotated objects – each annotation can only belong to digital objects which existed before (Agosti and Ferro, 2007; Agosti et al., 2007a; Agosti and Ferro, 2005), and fragments are younger than the object they belong to, since they are created during annotation. The age of a digital object can be determined by the hasCreationTime property which assigns each object a unique time stamp.

Constraint 3 (Temporal order of annotation hypertexts): Let t denote the time an item was created in the sense of a timestamp, i.e., the higher $t(o)$, the younger object o. Then

$$(a, b) \in E \Rightarrow t(a) > t(b).$$

□

The temporal order makes sense for the annotation hypertext. Let again be $e = (n, m) \in E$. If $l(e) =$ "hasAnnotationTarget", then n annotates m. It is impossible to annotate an object which does not exist yet, so consequently m must be older than n and it is $t(n) > t(m)$. The same holds true if $l(e) =$ "references" – it is only possible to reference older objects, so again it is $t(n) > t(m)$. If a fragment of an annotatable object was created in the annotation process, this object needs to exist. This means for $l(e) =$ "isFragmentOf", it is $t(n) > t(m)$ as well.

Proposition 1 (Annotation hypertext acyclic): *Each annotation hypertext $H = (V, E)$ is loopless (i.e., there is no edge whose start and end vertex are the same) and acyclic.*

Proof. Loopless: This is a consequence from Constraint 1.

Acyclic: Let us assume the annotation hypertext H is cyclic. Then there exists a path $(v_1, \ldots, v_n, v_1)$ in H, with $(v_i, v_{i+1}) \in E$ for $i = (1, \ldots, n-1)$. Because of Constraint 3, it is $t(v1) > ... > t(v_n)$, but also $t(v_n) > t(v_1)$, which is a contradiction. ■

3.2.2 Structured Annotation Hypertext

The annotation hypertext may not only contain documents and annotations, but also subcomponents in case we deal with structured documents and annotations. Therefore we extend our notion of an annotation hypertext with document structure.

Before we start, we have to define the temporal completeness of structured objects w.r.t. annotation, which means that a structured object being annotated, referenced or annotating cannot be changed after the act of annotation[5].

Constraint 4 (Temporal completeness of structured objects): Let x be a structured object (document or annotation) and SUB_x be the set of its subcomponents which are directly or indirectly connected with x through the isPartOf property. From a temporal perspective, we regard x as being *complete w.r.t. annotation* in the sense that no subcomponent was added to SUB_x after x annotates another object, or any $s \in SUB_x$ has been annotated or referenced by another item. If again t denotes the time an item was created, let y be the *youngest component* of x, i.e., $t(y) = \{\max(t(i)) | i \in \{x\} \cup SUB_x\}$, and o be the *oldest component* of x, i.e., $t(o) = \{\min(t(i)) | i \in \{x\} \cup SUB_x\}$. Then we define the following conditions:

[5]There might be cases where this assumption is violated, for example when annotations are used in an authoring process. After annotating a part of the document in creation, it might be extended by a new subcomponent. However, in our model we would regard the new extended version as a new document which is not annotated yet. Versioning in an annotation environment is a problem in its own right which is not further addressed here.

1. $t(a) > t(y)$ for each $i \in \{x\} \cup SUB_x$ with hasAnnotationTarget(a,i) or references(a,i); each object annotating or referencing x or one of its subcomponents must be younger than the youngest component;

2. $t(f) > t(y)$ for each $i \in \{x\} \cup SUB_x$ with isFragmentOf(f,i); each fragment of x or its subcomponents must be younger than the youngest component;

3. $t(o) > t(u)$ if hasAnnotationTarget(x,u) or references(x,u); if x annotates or references another annotatable object u, the oldest component must by younger than u. (Any $s \in SUB_x$ can be an annotation target, but not be an annotation itself.)

Let $u, v \in \{x\} \cup SUB_o$. The above conditions prohibit that $t(u) > t(i) > t(v)$ with $i \notin \{x\} \cup SUB_x$ annotating or referencing v or a fragment of v. In this case, u would have been added to $\{x\} \cup SUB_x$ after a component or subcomponent was annotated. u would be younger than i, which would violate condition 1. Similarly, in case i is a fragment of v, we would have condition 2 violated. Finally, if u annotates i or a fragment of i, then condition 3 is violated.□

We define the structured annotation hypertext as an extension of annotation hypertexts dealing with structured documents and annotations. We can create a structured annotation hypertext from annotation hypertexts by adding the structural relationships between components which are given by the isPartOf property.

Definition 2 (Structured Annotation Hypertext):
A *structured annotation hypertext* $SH = (V', E')$ is an extension of an annotation hypertext $H = (V, E)$ and created as follows:

- $V' = V, E' \supseteq E$
- If isPartOf(n, m) then $(n, m) \in E'$ and $l((n, m)) =$ "isPartOf".

Note that H is a subgraph of SH. $V' = V$ because all digital objects, even components, are vertices in annotation hypertexts. An annotation hypertext is thus a structured annotation hypertext without "isPartOf" edges.

Example 1 (Structured annotation hypertext): Figure 3.2 shows an example of a structured annotation hypertext. Here, document d1 is a structured document with subcomponents s1 and s2. The annotation a1 annotates s2. f is a fragment of d1 which is annotated by a3. a3 annotates both f and a1, and is a structured annotation containing t. a2 annotates s2 and references it to d2, thus creating a link from s2 to d2. □

In (3.6) on page 28 we defined structured annotations and documents by means of components, which are objects having exactly one isPartOf property. The problem is that a situation as it is depicted in Figure 3.3 still can occur – s2 is part of s1, s3 is part of s2, but s1 is part of s3, so we have a cycle here and a set of components from which none of them is a subpart of a document or annotation. This is certainly not what we want, so we have to ensure that in a set of connected components, there exists one component which is part of an annotation or document. If this is the case, each subgraph consisting only of "isPartOf" relations is a tree.

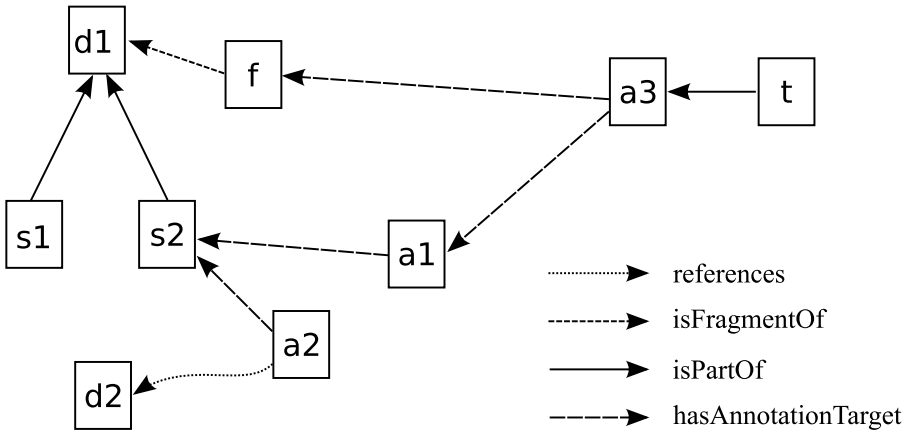


Figure 3.2: Example of a structured annotation hypertext

Constraint 5 (Structured objects are trees): Let

$$AN(x) = \{y | \mathsf{component}(y) \wedge (\mathsf{isPartOf}(x,y) \vee (\mathsf{isPartOf}(x,y') \wedge y \in AN(y')))\}$$

be the set on ancestor components of a component x. Either x or one of its ancestor components must be part of a document or annotation:

$$\forall x : \mathsf{component}(x) \Longrightarrow \exists y \in \{x\} \cup AN(x) : \mathsf{isPartOf}(y,y') \wedge (\mathsf{Annotation}(y') \vee \mathsf{Document}(y'))$$

□

Applying this constraints forbids the structure in Fig. 3.3.

We also forbid that a component or subcomponent of an annotation can annotate or be annotated by itself:

Constraint 6 (Self annotation and reference): Let a be a structured annotation and SUB_a be the set of its subcomponents which are directly or indirectly connected with a through the isPartOf property. No element from SUB_a can be an annotation, and Constraint 1 prohibits that a annotates itself. Let $SH = (V, E)$ be the structured annotation hypertext containing a, so that $a \in V$. Then there must not exist any edge $(a, n) \in E$ with $n \in SUB_a$. We also have to take into account the case that a references or annotates a fragment of its subcomponents. So we say that there must not exist an edge $(a, n) \in E$ and $(n, m) \in E$ with $l((n, m)) =$ "isFragmentOf" or and $m \in SUB_a$. Figure 3.4.a illustrates some forbidden cases of self annotation. □

Note that Constraint 3 alone would not prohibit self annotation.

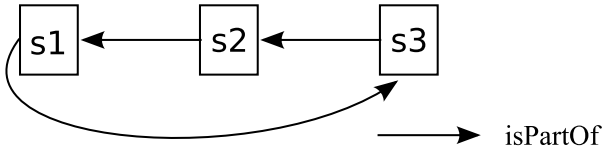


Figure 3.3: Cyclic component structure

(a) (b)

Figure 3.4: (a) Three cases of self annotation: Through an annotation target, a reference and indirectly through a fragment f.
(b) Sketch of a cyclic structured annotation hypertext; dotted line denotes edge/path in H, solid line edge/path in S.

Proposition 2 (Structured annotation hypertext acyclic): *Each structured annotation hypertext $SH = (V, E)$ is loopless and acyclic.*

Proof. Loopless: This is again a consequence from Constraint 1.

Acyclic: We prove that SH is acyclic by showing that a cyclic SH would violate temporal constraints. We create the subgraph $S = (V', E')$ from SH as follows: $(n, m) \in E'$ iff $l((n, m)) =$ "isPartOf". $n \in V'$ and $m \in V'$ iff $(n, m) \in E'$. So S only consists of part-of relations and represents structured documents and structured annotations, but no connections between them. S is acyclic, since otherwise Constraint 5 would be violated.

Let $H = (V'', E'')$ be the annotation hypertext from which SH was created. H contains only edges which are not contained in S, and it is $E = E' \cup E''$ with $E' \cap E'' = \emptyset$. We know from Proposition 1 that H does not contain any loops and is acyclic. Let us assume SH is cyclic, and $p = (v_1, \ldots, v_i, v_{i+1}, \ldots, v_n, v_1)$ is a cycle in SH. Due to the fact that both S and H are acyclic, this path must contain at least one edge from S and one from H. Without loss of generality, let $(v_i, v_{i+1}) \in E'$ be the first edge of p in S. Then v_i and v_{i+1} must be subcomponents of an object a (or $v_{i+1} = a$, respectively). In order to create a cycle, a must be an annotation, since otherwise a would only contain incoming edges. Furthermore, there must exist a vertex $o \in p$ which is connected to v_i through an edge in H. Figure 3.4.b illustrates the situation. There are two possible cases:

- $o = a$, then a does a self annotation, which is not allowed due to Constraint 6;
- $o \neq a$. Then all objects on the path from a to o must be older than a, thus $t(a) > t(o)$ due to Constraint 3. Since o annotates v_i or a fragment of v_i, it must be $t(o) > t(v_i)$, so $t(a) > t(o) > t(v_i)$, which violates Constraint 4, since a would have been created after v_i was annotated. ■

3.3 Summary and Discussion

The output of this chapter is a formal model describing the objects, concepts and relations of the annotation universe which we deem important in an annotation scenario. The model is based

on existing ones addressing the objects in digital libraries (Gonçalves et al., 2004), annotations of digital content (Agosti and Ferro, 2007) and structured document retrieval (e.g., (Fuhr et al., 2002; Fuhr and Großjohann, 2004; Chiaramella et al., 1996)). We apply Description Logics to describe our model; the T-Box shows the relations between certain classes like digital objects, fragments, annotations and documents, whereas the A-Box specifies the individuals. On this level we identified the structured annotation hypertext and showed that it is acyclic.

The model presented in this chapter can be used as the underlying data structure to design future annotation-based retrieval methods, combining structured documents with annotation hypertexts. It integrates, sometimes extends or weakens, many concepts, relationships and constraints found in the different models mentioned above. The proposed model does not claim to be yet another model of annotations and digital libraries besides the ones already mentioned, but emphasises the objects which we deem useful and which should be considered for retrieval involving annotations. How these objects are used for retrieval is subject of the next part of this thesis, and if exploiting them really improves retrieval effectiveness is discussed in Part III. It is understood that this model should not be seen as carved in stone; instead, it should be regarded as an open model and may be tailored, if necessary, to a specific annotation-based retrieval application.

We are now ready to introduce the main contribution of this thesis, which is a probabilistic, object-oriented, logic-based framework for annotation-based retrieval. The framework, its functionality, syntax, semantics and implementation is discussed in the next part of this thesis. It operates on a structured annotation hypertext, like the one discussed in this chapter, as the underlying data structure.

Part II

The POLAR Framework

4

Annotation-based Knowledge Modelling and Retrieval with POLAR

> The best defense against logic is ignorance.
>
> *(Blaise Pascal)*

In Part I of this thesis we discussed annotations and their peculiarities, as well as identifying objects which we deem important for annotation-based retrieval. The outcome of that part is the definition of the structured annotation hypertext, which is the main data structure upon which annotation-based IR can be performed.

But what is "annotation-based IR"? How can we provide a framework which is able to model structured annotation hypertexts and, exploiting the rich information we find there, satisfy many information needs arising when dealing with annotations in a search scenario? How can we actually perform "annotation-based retrieval"? While the Description Logics used in the last chapter (and also derivates of it, like the Web Ontology Language (OWL)) is a feasible language to describe the objects and relationships in the annotation universe (and we used them with no other intention in mind), we need a more sophisticated representation and description of these objects and their relationships to perform an actual annotation-based retrieval task, especially to cover concepts like (probabilistic) term weights and inverse document frequencies, as they are commonly known from various IR approaches. So the main topic of this chapter is to introduce a framework, called POLAR, which is able to model structured annotation hypertexts on the one hand, and to employ the information contained in this representation for annotation-based retrieval on the other hand. The discussion of POLAR in this and the two subsequent chapters shall answer the "how" questions – how to model structured annotation hypertexts and how to perform annotation-based retrieval in order to satisfy (possibly sophisticated) information needs related to annotation.

This chapter starts by giving an overview of information retrieval and some of its important models. This overview also contains a general discussion of what annotation-based retrieval is about. Afterwards, the POLAR framework is presented informally. POLAR is the tool to model structured annotation hypertexts, query them and perform annotation-based IR by applying probabilistic inference and implication. To show how POLAR can support information needs which also might involve non-topical evidence, further application showcases are presented. After that, Section 4.4 discusses some related work and positions POLAR in the

context of hypertext and structured document IR, discussion search and of course annotation-based retrieval. The subsequent section concludes this chapter and discusses its main findings.

4.1 Information Retrieval

This section gives an overview about information retrieval. It commences with an introduction to the field and then presents important retrieval models. Since in POLAR we deal with hypertexts and structured documents, these topics are subsequently introduced. Finally, the relatively young field of annotation-based IR is presented.

4.1.1 Introduction

According to Baeza-Yates and Ribeiro-Neto (1999), *information retrieval* (or IR for short) "deals with the representation, storage, organization of, and access to information items. The representation and organization of the information items should provide the user with easy access to the information in which he is interested". The main goal of IR is thus to let the user easily access the information, stored in information items pooled into a collection, he or she needs to satisfy an *information need* which arises when fulfilling a certain task. So we have an information need on the one hand, and a set of information items (usually called *documents* in the IR context) containing relevant information on the other hand. In fact, finding *relevant* information for a given information need is a difficult task, as Mizzaro (1998) describes. The goal is to deliver the user exactly the information which is relevant to the user's real information need at the given time, to the given topic, to fulfil the given task within a specific context. Today's IR systems and search engines can support this goal only to a certain degree. Problems arise, for example, in describing the actual information need and to transform it into a (keyword-based) query language the IR system understands. On the other hand, many IR systems see documents as atomic units and return them instead of the single piece of information which is relevant. Users may wonder why a document was retrieved and need to find the relevant information within the document. Instead of returning the *information* relevant to a *real information need*, contemporary IR systems usually return *documents* which are relevant to a *query*.

We can distinguish between two different views on the IR problem (Baeza-Yates and Ribeiro-Neto, 1999, p. 7): the *computer-centred view* focuses on building efficient indexes and developing algorithms which process a query efficiently and effectively, i.e. they should return a high-quality ranking of documents w.r.t. the query as quickly as possible. In contrast to that, the *human-centred view* studies the user's behaviour, tries to understand his or her information needs and how retrieval systems can be operated to best satisfy the information need. The computer- and human-centred view are not mutually exclusive; in fact, query processing as studied in the computer-centred view is an important strategy and may be part of a bigger solution to satisfy information needs. When talking of IR, usually the computer-centred view is meant. The human-centred view is sometimes referred to as *information seeking and searching*. In this thesis, we mainly focus on the computer-centred view of IR (and use the term "IR" for this view), bearing in mind that the results of the work presented here might as well be interesting in the human-centred view. In fact, the approach presented later is aimed at providing a flexible framework which can be operated to support sophisticated information needs.

4.1.2 An Overview of Retrieval Models

We give an overview of the most important IR models and concepts. We begin with a general-purpose conceptual model and then discuss non-probabilistic and probabilistic models, and models based on uncertain inference and probabilistic logics.

4.1.2.1 Conceptual Model

Fuhr (1992) introduces a *conceptual model of IR*, which is depicted in Figure 4.1. In this model, Q and D are the sets of possible queries and documents, respectively. Documents can be judged relevant or non-relevant to a query; this is done by providing *relevance judgements*, which are usually given by users or domain experts. Relevance judgements are usually based on information needs rather than queries (which are representations of information needs), but here we assume that documents are judged relevant w.r.t. a query. If we assume a binary relevance scale, the set $\mathcal{R}$ of relevance judgements is $\mathcal{R} = \{R, \overline{R}\}$ with R saying that the document is relevant to the query, and $\overline{R}$ saying that it is not. Relevance judgements can thus be regarded as a mapping $r : Q \times D \to \mathcal{R}$.

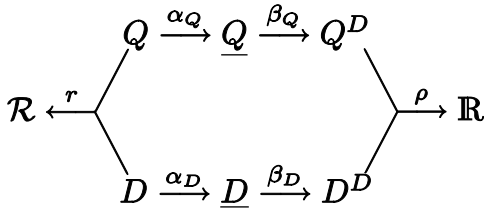


Figure 4.1: Conceptual Model of Information Retrieval (Fuhr, 1992)

To process queries and documents, both must be transformed into an internal representation and description which is machine-processable according to the given retrieval function. The function $\alpha_Q : Q \to \underline{Q}$ transforms queries into their *query representation*. α_D does the same for documents and leads them into their *document representation*. Very often a second transformation for queries and documents is necessary, which converts queries into their *query description* and documents into their *document description*. $\beta_Q : \underline{Q} \to Q^D$ and $\beta_D : \underline{D} \to D^D$ are the corresponding functions for queries and documents, respectively. The process of *document indexing* usually applies both α_D and β_D consecutively to create the document index. For instance, in many approaches, α_D transforms a full-text document into a *bag of words* by removing punctuation and frequently occurring words (so-called *stop words*, which, in English, are words like "and", "or", "is", "a" etc.), and additionally applying stemming algorithms in order to use word stems in place of the words themselves (like "comput" instead of "computing"). The function β_D might then take this bag of words and calculate *term weights* based on, e.g., the frequency of a term within the document. Documents are then often described as vectors of terms, where the ith element of the vector contains the weight of term t_i in the document. α_Q and β_Q might perform similar transformations on queries. What document and query descriptions look like and how the corresponding transformation functions are defined depends on the retrieval model.

To process queries and return a ranked list of documents which are relevant w.r.t. a query, a *retrieval function* $\rho : Q^D \times D^D \to \mathbb{R}$ is applied; this function operates on query and document descriptions and returns a *retrieval status value* (RSV) $rsv \in \mathbb{R}$ for each document. The output of ρ is used to create a *ranking* of documents according to decreasing retrieval status values.

4.1.2.2 Non-Probabilistic Models

Several retrieval models exist which can be described by means of the conceptual model (Baeza-Yates and Ribeiro-Neto, 1999, chapter 2). In the *Boolean model*, documents are described as term sets. If there are T terms in the collection, the document description is a T-dimensional vector $\vec{d}$ with $d_i \in \{0, 1\}$ and $d_i = 1$ if t_i appears in d, and 0 otherwise. Query descriptions are Boolean expressions like "$t_1 \wedge (t_2 \vee \neg t_3)$". The Boolean retrieval function ρ returns documents which match the Boolean expression.

The *vector space model* (VSM) (Baeza-Yates and Ribeiro-Neto, 1999, section 2.5.3) is a retrieval model which was first introduced by Salton when working on the SMART project. It recognises that the use of binary weights and the sharp partitioning into matching and non-matching documents is too limiting. Therefore, documents and queries in the VSM are described as vectors of (negative or positive) term weights in a T-dimensional vector space. The retrieval function ρ determines the similarity between the document and the query vector, for instance by calculating the cosine of the angle between query and document vector, or the scalar product, so that $\rho(d^D, q^D) = \vec{d} \cdot \vec{q}$ is the RSV of d w.r.t. q. The system returns a ranking of documents according to decreasing RSVs.

To determine term weights, it is common in IR to use statistical values like the *term frequency* tf_{ij} of a term t_i in a document d_j. tf_{ij} is usually normalised by the document length (and denoted ntf_{ij} then). A simple example is $ntf_{ij} = tf_{ij} / \max_l(tf_{lj})$ where $\max_l$ is the frequency of the most frequent term in the document. Another important component is the *inverse document frequency* idf_i which is calculated upon the number of documents t_i appears in. For example, $idf_i = \log(N/n_i)$ with N as the number of documents in the collection and n_i as the number of documents t_i appears in. The motivation behind this factor is the assumption that terms which appear in many documents are less discriminatory than those appearing in fewer documents. Therefore, *idf* increases the more rare a term is. Many retrieval approaches based on the VSM balance the two factors *tf* and *idf*, for example by using $w_{ij} = ntf_{ij} \cdot idf_i$ as the weight for t_i in d_j. The document d_j is then described by the vector $\vec{d_j} = (w_{1j}, \ldots, w_{Tj})$. We call such approaches *tf* $\times$ *idf*-based methods.

4.1.2.3 Probabilistic Models

The aim of most *probabilistic models* (Crestani et al., 1998; Fuhr, 1992) is to rank documents in decreasing order of $P(R|q, d)$, the *probability of relevance* of a document d with respect to the query q. The *Probability Ranking Principle* (PRP) gives a theoretical justification for creating a ranking based on this probability (Robertson, 1977). Probabilistic relevance models thus estimate the probability $P(R|q, d)$, often by applying Bayes' Theorem and making certain independence assumptions. We can roughly distinguish between model-oriented and description-oriented approaches. The former are based on some probabilistic independence assumptions.

Model-oriented approaches can be categorised into query-related and document-related learning (Fuhr, 1992). An example for a query-related approach is the *binary independence retrieval* model (BIR) (Robertson and Sparck Jones, 1976) which utilises relevance feedback[1] data to reweight search terms of a given query q. In BIR, a document d is represented as a T-dimensional term vector $\vec{x}$, so that $P(R|q, d)$ becomes $P(R|q, \vec{x})$. In contrast to that, the *binary independent indexing* (BII) approach (Fuhr and Buckley, 1991) is query independent but document dependent. In the BII model, two probabilities have to be estimated: $P(R|d)$, the probability

[1] Relevance feedback is the process of judging documents in a ranking as relevant or non-relevant.

that d will be judged relevant to an arbitrary query, and $P(R|t_i, d)$ the probability that d is relevant given it contains the index term t_i.

The idea of *description-oriented approaches*, which have their roots in pattern recognition, is to apply a learning strategy for document indexing which is based on term *features* in documents (in contrast to the term itself). A description-oriented method was developed in the Darmstadt Indexing Approach (DIA) (Biebricher et al., 1988). Like in the BII model, the aim is to index documents probabilistically, but instead of estimating $P(R|t_i, d)$, the probability $P(R|\vec{x}(t_i, d))$ is calculated. $\vec{x}(t_i, d)$ is a feature vector which contains values of attributes of t_i and d. Such attributes may be the within-document frequency of t_i in d, the inverse document frequency of t_i in the collection or hints about the location of t_i (for instance if the term appears in the document title). An indexing function $e(\vec{x}(t_i, d)) \approx P(R|\vec{x}(t_i, d))$ is derived by means of linear (Fuhr and Buckley, 1991), polynomial (Fuhr and Buckley, 1993) or logistic (Cooper et al., 1992; Fuhr and Pfeifer, 1991) regression. For each term t_i in d, a term weight is then computed as $e(\vec{x}(t_i, d))$. The description d^D of a document d consists of a vector of such term weights. Given that the query is described as a term vector as well, the scalar product can deal as a retrieval function (like in the vector space model).

The probabilistic models so far have in common that they rely on knowledge about the set R of relevant documents in order to estimate $P(R|q, d)$. A probabilistic model which does not need relevance judgements is the *2-Poisson model*, which was formulated by Bookstein and Swanson (1974) first. The aim of the 2-Poisson model is to decide whether an index term t_i should be assigned to a document d_j or not. The underlying assumption is that tf_{ij}, the number of occurrences of a term t_i within a document d_j, is distributed differently within these two classes, according to a Poisson distribution. The required parameters to acquire a probabilistic indexing term weight can be estimated from the document collection.

The 2-Poisson model has been the starting point for several other (probabilistic) indexing approaches. One example are the BM-style term weighting functions (Robertson and Walker, 1994), from which the famous BM25 function (Robertson et al., 1995) is derived. Ponte and Croft (1998) propose a *language modelling* approach which does not rely on relevance judgements, but only uses statistics coming from the collection itself. The idea behind language models is to calculate $P(q|m_d)$, which is the probability that the query q can be generated from the language model m_d underlying the document d. The calculation of this probability employs $P(t|m_d)$, which is the probability that the term t can be derived from d's term distribution. It is then $P(q|m_d) = \prod_{t \in q} P(t|m_d) \cdot \prod_{t \notin q}(1 - P(t|m_d))$. Problems arise from the first product when no query terms appear in the document. In that case, $P(t|m_d)$ is approximated based on collection statistics.

Another model which can be regarded as inspired by the 2-Poisson model is the *divergence from randomness* approach (Amati and van Rijsbergen, 2002), which is related to language models. Term weighting is performed by measuring the divergence of the actual term distribution from that obtained under a random process. The basic idea is to compute the term weight as the product of two functions Inf_1 and Inf_2, which measure the informative content of a term. Inf_1 utilises the probability that term t has tf occurrences in document d by pure chance. The lower this probability, the higher the informative content of the term. Inf_1 can be based on certain models of randomness. Inf_2 is related to the risk of accepting the term as a good descriptor for the document and measures the information gain which can be achieved with this term. Inf_2 is regarded as a normalisation of Inf_1. The resulting product $w = Inf_1 \cdot Inf_2$ for a term t_i can be used in a term vector $\vec{d}$ describing the document d.

4.1.2.4 Models Based on Uncertain Inference

A paradigm which is related to probabilistic models is the view of *information retrieval as uncertain inference* (Crestani and Lalmas, 2001). In these logical models of IR, it is assumed that documents and queries can be described as a set of logical formulae. One of the strengths of this approach is that additional knowledge, for example coming from a thesaurus or an ontology, as well as metadata and any other knowledge can easily be integrated into the retrieval function. Since classical logic might not be adequate to represent queries and documents, many logic-based models apply a logic for uncertain inference and try to estimate the *implication probability* $P(d \rightarrow q)$ that a document d implies a query q, which was first proposed by van Rijsbergen (1986). Sloppily speaking, $P(d \rightarrow q)$ measures the amount of information we have to add to let $d \rightarrow q$ become true. One example of a possible interpretation and application of $P(d \rightarrow q)$ for retrieval is the imaging approach reported in van Rijsbergen (1989). Another formalism which can be used for the calculation of $P(d \rightarrow q)$ is provided by *Bayesian inference networks* (Turtle and Croft, 1990).

Many standard retrieval models like the Boolean or the vector space model can be expressed in terms of probabilistic inference, as Wong and Yao (1995) show. They apply an epistemological view on probabilities and define the implication probability as a conditional one, namely $P(d \rightarrow q) := P(q|d)$. Depending on certain independence and disjointness assumptions on the underlying concept space, several retrieval models can be expressed as uncertain inference. Other considerations aim at mapping the implication probability $P(d \rightarrow q)$ to the probability of relevance $P(R|q, d)$ (Nottelmann and Fuhr, 2003; van Rijsbergen, 1992) so that the PRP gives a theoretical justification to rank documents with respect to decreasing implication probabilities.

Besides models which estimate $P(d \rightarrow q)$ directly, there are other retrieval frameworks which use the notion of probabilistic inference for retrieval tasks. Two examples are terminological logic and probabilistic Datalog. One of the advantages of these frameworks is that they offer means for complex document and query representations and descriptions, and allow for the integration of additional knowledge into the retrieval process. Meghini et al. (1993) describe an IR framework based on *terminological logic* called MIRTL, which has its roots in knowledge representation and semantic networks. Documents are described as a set of concepts and roles, and the actual document is an instance of its corresponding concept. Queries in MIRTL are described as concepts and roles as well; when evaluating a query, the inference engine returns all individual constants whose concepts and roles are subsumed by the query description.

Probabilistic Datalog (PD) (Fuhr, 2000) is the probabilistic extension of Datalog based on Horn clauses. Documents and queries can be represented as facts and rules. Each fact can be assigned a probability. Rules create new intensional knowledge by combining facts and their corresponding probabilities with respect to the rules of probability theory. Due to its expressiveness, PD is a powerful tool to model advanced retrieval functions and frameworks which are also capable of integrating additional knowledge. For example, POOL (Rölleke, 1998; Fuhr et al., 1998) is a framework for object-oriented structured document retrieval which is based on PD. We will learn more about PD and POOL in the remainder of this thesis.

There are certain other models utilising (probabilistic) logics in many different ways. For example, Müller and Thiel (1994) use abductive inferencing for query expansion and interactive IR. Further models are reported in Crestani and Lalmas (2001).

4.1.3 Exploiting the Neighbourhood – Hypertext, Structured Document and Web Retrieval

The retrieval and indexing models discussed so far regard documents as atomic units – no relations between documents or within them are considered. But the emergence of hypertext systems in the late 80s and of course the World Wide Web (WWW) made it possible to broaden the view and see documents and document fragments embedded in a linked environment.

According to Agosti (1996), a *hypertext* is composed of *nodes* and a *network of links*. Nodes may be document fragments, but also single documents. In the former case, links between these nodes represent intra-document relations like the document structure; in the latter case, links between nodes are inter-document relations, for example to similar documents. Within the WWW we usually find a mixture of intra- and inter-document links. Links can be classified in different link types, like structural links (reflecting the document structure), referential links (e.g. between a document and a document that is citing it) and further unspecified associative links (Agosti and Melucci, 2000). There can be numerous other link types in a system (Trigg, 1983, chapter 4). Nodes may represent different kinds of digital media (like videos, images, sound); to reflect this, the term *hypermedia* is commonly used.

One of the main differences between hypertext and standard document collections is the additional possibility to realise information search by *navigation* and *browsing*. Relevant documents are not only found by examining a linear ranking delivered by a typical search engine, but also by following links pointing to potential other relevant nodes in the hypertext. To this end, classical IR methods can aid navigation and browsing on the one hand, and searching on the other hand. They can aid navigation and browsing by giving hints and links to next relevant passages (Hammwöhner and Thiel, 1987) or automatically construct new hypertext links of different type like similarity links (Agosti and Melucci, 2000). IR methods can further help to find *entry points* in the hypertext which are a good starting point for further navigation. Neighbouring nodes and their corresponding (possibly typed) links (e.g. established by citations) can be exploited in order to calculate the relevance of a node. This way, the hypertext defines the context a node/document is embedded in. The interested reader is referred to Agosti and Smeaton (1996) for a thorough discussion of hypertext retrieval including references to many approaches.

A very special case of hypertexts are *structured documents*. Nodes in structured documents are document fragments which are connected by structural links. An example is a book which is made up of chapters, sections, subsections and so on. A chapter node, for instance, may point to several section nodes, which again are linked to subsection nodes. Structured documents basically form trees. The *logical structure* of documents is nowadays represented using the eXtensible Markup Language (XML). The success of XML as a means to exchange formatted data (like database records) on the one hand, and for the representation of structured documents (containing text, multimedia and metadata) on the other hand motivated the creation of new models and methods for *XML information retrieval* (XML IR). Like many classical hypertext approaches, XML IR methods exploit the structural context of nodes to allow for more precise access to the relevant content (by trying to find a best entry point within the document structure). The fact that, in contrast to HTML, nodes in an XML tree usually convey defined semantics, allows for the definition of complex retrieval tasks which are not only able to process traditionally content-oriented queries, but can also handle hints about the type of elements to be retrieved (so called *content-and-structure queries*). The success of XML also meant that test collections for structured document IR became available. These led to the foundation of the Initiative for the Evaluation of XML Retrieval (INEX), whose goal is not only the compara-

tive evaluation of XML IR approaches within defined tasks, but also to discuss new evaluation measures which take into account the peculiarities of structured documents (see, e.g. (Fuhr et al., 2007)).

Another special case of hypertext information retrieval is Web IR. As the name suggests, Web IR is tailored to the WWW and documents composed in the Hypertext Markup Language (HTML). In many Web IR approaches, a link analysis is performed to compute non-topical values, for example to decide if a document is a good authority (when many Web pages point to it (Page et al., 1998; Kleinberg, 1998)) or a good hub (when the document points to many good authorities (Kleinberg, 1998)). Other approaches apply methods known from traditional hypertext IR (see, e.g., Savoy and Picard (1999)) or create and exploit a rich document representation which comprises, for instance, the full text and the headings found in Web documents, but also the anchor text of referring Web pages (Craswell and Hawking, 2003).

4.1.4 Annotation-based Retrieval

Annotation-based IR is a relatively young research area compared to hypertext and XML IR. We therefore introduce this area first, by describing main retrieval scenarios and its relation to hypertext and structured document retrieval. Possible benefits of annotation-based IR are discussed afterwards.

4.1.4.1 Main Scenarios

As we have seen in Chapter 3, the main data structure we are dealing with is a hypertext, comprising structural links (like in XML documents) and intra-document links to fragments as well as inter-document ones to annotations or other documents. Having given an overview of the relevant areas, in particular information retrieval, hypertext IR and XML IR, we can now discuss what we actually mean by "annotation-based retrieval". Based on the structured annotation hypertext as defined in Chapter 3, we can identify the following two main scenarios for annotation-based retrieval (cf. Frommholz (2005b)):

1. *Document search*: Users are only interested in the main documents of the repository. In this scenario, annotations are just auxiliary documents and are not equal to the main documents. Nonetheless, since annotations contain information about the annotated documents (as meta annotations) or extensions to its content (as content annotations), annotation-based document search will exploit the context given by the annotations and annotation threads attached to a document.

2. *Annotation and discussion search*: In this scenario, annotations are the main target of the search – users are interested in finding relevant annotations. Finding relevant annotations can, for instance, uncover interpretations, thoughts and ideas not present in the main documents, or reveal the motivation behind a decision. As we have seen in Chapter 2, annotations can be used to model discussions and discourses. In the discussion search scenario, users are interested in parts of or the whole annotation thread satisfying their information need. Systems should hint them to suitable entry points in the discussion thread.

The scenarios reflect the dualism between annotations as content enrichment and as stand-alone documents (see Section 2.2.2). Both scenarios do not exclude each other, but may be combined, for example if users do not distinguish between annotations and main documents,

but just want to retrieve any relevant object in the repository. Since the desired objects are embedded in a context given by their linked items, annotation-based IR is a kind of *context-based retrieval*. Due to its underlying data structure, annotation-based IR is strongly related to hypertext (or hypermedia) IR as well as to structured document retrieval or XML IR[2].

4.1.4.2 Relation to Hypertext and Structured Document IR

Like the structured annotation hypertext is a special kind of hypertext, annotation-based IR can be viewed as a special kind of hypertext IR in which we distinguish between different basic objects, the (main) documents and the annotations. Known hypertext retrieval approaches do not take the peculiarities of annotation hypertexts into account, so they are, although in general applicable, usually too generic for the given problem.

Annotation-based IR can also be regarded as an extension to structured document retrieval or XML IR, since it does not only consider the internal logical structure of documents, but also external objects attached to them (the annotations). Classical XML IR methods would cover only a subset of the possible retrieval tasks when dealing with structured documents and annotations. We see from this discussion that annotation-based IR indeed establishes a new class of retrieval tasks and problems to solve.

4.1.4.3 Possible Benefits

What do we hope to gain from annotation-based IR? One possible benefit is that it potentially addresses the *vocabulary problem*. Usually, retrieval approaches assume that the document author and the user share the same vocabulary, which is then adapted by the (automatic) indexer. This assumption does not always hold – for instance, users and author might use different terms for the same facts or even circumscribe them (an expert in a specific field might use a different vocabulary than the common user). A fact possibly changes its describing terms after some time (this is especially true with the interpretation of historic texts). With annotations, a third player joins the game, the annotator. If we combine the evidence coming from documents and annotations, chances are higher that the user uses a term for a fact which either the document author or the annotator used. An example is the term 'land' in the Bible, which generally, but not always, refers to the land of Israel or Canaan (Fraenkel and Klein, 1999). So annotations can help to determine the relevance of passages where the term 'land' appears by associating them to Israel or Canaan, but also help to find the exceptions. As Fraenkel and Klein explain, a query for "war in Babylon" would only be successful when annotations are involved, since in a relevant passage[3] 'land' exceptionally refers to 'Babylon', which is only mentioned in an annotation coming from a Bible commentary. Another example comes from the COLLATE scenario and can be seen in Figure 2.1 on page13. Consider a censorship document which says that a film was censored for, say, morality reasons. The first annotator interprets the document in a way that she thinks the censorship reasons were actually political ones. This way, the document is associated with political censorship, although from its content there is no evidence for it. The document is probably relevant (together with its attached annotation) when film students search for political censorship. In the case that annotations express facts differently or associate things, they help to find more documents which would otherwise not be found, that is, they increase the *recall*. The danger, of course, is

[2] If we see XML IR detached from the framework it is named after as a variant of structured document retrieval, that is. We do not want to imply that annotations should be encoded in XML.

[3] "A sound of war is in the land, and of great destruction." (Jeremiah 50:22)

that through annotations, new terms are introduced to the annotated document which could actually not be associated with it. The quality of annotations is another important factor. Especially in public discussion forums, many comments lack the required quality to be taken seriously.

When terms occur in annotations, their targets or in annotated fragments, annotations might have a beneficial effect as well. For instance, automatic term indexing estimates the probability that a document can be indexed with a specific term. It usually does so by counting the number of occurrences of that term in the document and possibly applying some normalisation based on the document length. But an uncertainty remains that the automatic association of a document to an index term is not correct. If such an index term occurs both in a document and its annotations, this raises our certainty that we really can index the document with that term, since both document and annotation seem to talk about the same topics. The same holds if a term occurs in an annotated fragment, especially if the fragment is a direct highlighting. When terms occur in several subcontexts, the probability that the decision to index the document with that specific term increases. This could possibly boost the *precision* of our search.

As indicated above, the relationships between objects in the structured annotation hypertext contain additional information which, if interpreted correctly, can be beneficial for retrieval as well. If, for example, we can distinguish whether an annotation is positive or negative w.r.t. the annotated object, we are able to draw conclusions about the trustworthiness of comments in a discussion. Furthermore, if an article has many comments, it shows that the article's topic is somewhat popular (otherwise nobody would care about the article). We see that the amount of incoming or outgoing links in a structured annotation hypertext indeed contains evidence which is worth being considered. We also see that, besides the topical aboutness of annotations or documents, a structured annotation hypertext implicitly or explicitly contains further information which might be helpful to satisfy users' information needs. Some examples are discussed later in Section 4.3.

In the next section we are going to present the POLAR framework. The aim of POLAR is to support both annotation-based document search, and annotation and discussion search.

4.2 The POLAR Framework

As described by van Rijsbergen and discussed in Section 4.1.2.4, information retrieval can be seen as uncertain inference (van Rijsbergen, 1986). We follow this idea and present a probabilistic, object-oriented representation of structured annotation hypertexts as defined in Chapter 3 and discuss annotation-based querying and retrieval functions based on this model. The framework is called POLAR (Probabilistic Object-oriented Logics for Annotation-based Retrieval). Within POLAR, it is possible to pose queries to the knowledge base for retrieving documents (with the help of annotations) or relevant annotations. POLAR is similar to and very much influenced by POOL, an object-oriented, probabilistic logical model used for representing complex structured documents (Rölleke, 1998; Fuhr et al., 1998), but with a different focus.

POLAR was introduced in Frommholz and Fuhr (2006b). Since then, POLAR has undergone slight changes in its syntax. In the following, we extend the description given in Frommholz and Fuhr (2006b), reflecting its current syntax. We commence with a discussion of the motivation behind POLAR, its main features and how it compares to POOL. Subsequently, we show how documents and queries are represented and described in POLAR. We then introduce POLAR informally and illustrate how certain information needs are supported. A formal definition of

POLAR's syntax and semantics can be found in Chapter 5, its implementation is discussed in Chapter 6.

4.2.1 Motivation

4.2.1.1 Why a Probabilistic, Logic-based Framework?

In the discussion so far we have seen that annotations, annotated objects and their interrelations build a complex data structure. The motivation behind POLAR is to create a powerful tool which is able to support sophisticated information needs related to structured annotation hypertexts and annotation-based retrieval. To do so, it is not sufficient to adapt one of the classical retrieval approaches (like language models or the vector space model) to also consider the content of annotations in order to create a retrieval status value for an object w.r.t. a query. When serving sophisticated information needs, we might need to take into account different kinds of relationships between different kinds of complex objects. Thus, a tool for annotation-based retrieval should also allow for queries to the underlying knowledge base containing the structured annotation hypertext. Such queries could for example ask for the number, types and polarity of annotations made to a specific object, or for the scope of an annotation in order to return only annotations in a ranking which the user is allowed to see. A separation in content and meta level annotations is crucial in order to distinguish whether an annotation extends the content of the annotated objects or says something about them; both kinds of annotations can be treated differently when satisfying information needs. Annotation-based retrieval functions should also be able to take the logical structure of documents into account, which allows for annotation-based structured document retrieval where annotations are used to find a best entry point within the document structure. External information, for example coming from a thesaurus, may be incorporated into the retrieval function. Furthermore, especially annotation-based discussions often contain contradictions (recall the COLLATE example in Section 2.1.2.2 and the disagreement w.r.t. "political reasons"). It would be desirable that annotation-based retrieval methods could handle these situations. These considerations should make clear that we need a rich representation and description of structured annotation hypertexts and the content of annotations and documents, and we need the means to exploit this representation for annotation-based retrieval.

How can we provide a tool which offers such a rich representation and the whole range of functionality discussed above? In IR, logic-based approaches like probabilistic Datalog (Fuhr, 2000) are general and flexible frameworks to set up complex retrieval functions which exploit various kinds of evidence. Such frameworks, in this case based on predicate logics and probabilistic inference, provide users with a huge degree of freedom to model the underlying data structures and suitable retrieval functions. However, these models and functions can easily lead to inconsistent and inefficient Datalog programs. Probabilistic Datalog is very generic, and it would therefore be desirable to have a conceptual model which is tailored to structured annotation hypertexts. Such a model could enable users to easily model all the objects, their content and interrelations in structured annotation hypertexts (without bothering about how to store them). It would provide them with core annotation-based retrieval functions on the one hand, and the ability to enrich this core functionality accordingly on the other hand. For structured documents, POOL (Rölleke, 1998) is such a framework. The idea is to use POOL as a starting point and extend it to be able to cope with structured annotation hypertexts.

4.2.1.2 Going Beyond POOL

POOL is s framework to model structured, complex objects. Its object-oriented design features provide some helpful mechanisms like the possibility to describe complex objects through propositions, classifications and attributes. An example POOL program (taken from Rölleke (1998)) shall illustrate this. It models a complex, structured document d1 having the two subparts s1 and s2. It also shows some interesting feature POOL provides, namely four truth values with an open world assumption. Consider the following POOL object:

```
d1[ 0.9 s1[ 0.8/0.2 sailing ]
    0.7 s2[ 0.6/0.4 sailing ]]
```

The document d1 consists of the two subparts s1 and s2; these are accessed with 0.9 and 0.7 probability, respectively. In the context of s1, the term "sailing" is *true* with 0.8 probability, and it is *true* with a probability of 0.6 in the context of s2. In POOL, we can also specify the probability that propositions are *false* (and even the probability that they are *inconsistent*); in the example, "sailing" is *false* with 0.2 probability in s1 and with a probability of 0.4 in the context of s2. A query for documents about "sailing", expressed in POOL as

```
?- D[sailing]
```

would return a ranking containing s1 and s2, but also d1 due to an approach called knowledge augmentation which propagates the weights from the subcontexts s1 and s2 to the augmented context d1(s1,s2) for the document d1. Propositions in POOL cannot only be terms, but also classifications and attributes, making it a powerful tool to describe spatial relations as it would be useful for multimedia documents.

Tree-like annotation threads might be modelled in POOL so that each annotation is a subcontext of the object it annotates. Supporting structured multimedia objects is a desirable feature for annotation-based IR, as annotations and annotated objects can be multimedia documents as well – one could think of voice comments or even video annotations. Another interesting feature of POOL is its ability to deal with four-valued logics, which provides means to cope with inconsistent and contradicting knowledge – indeed, also with annotations, knowledge can get inconsistent and contradicting in case one annotator says that a proposition is true and another one states that the same proposition is false. Inconsistencies and contradictions naturally arise in annotation-based discussions. Additionally, the *open world assumption* supported by POOL says that if there is no evidence that a proposition is true, we cannot infer that it is false (like we would with a *closed world assumption*). This is an interesting feature for document indexing, because the lack of an index term for a document does not mean that it must not be indexed with that term. POOL can also create new intensional knowledge by means of rules, and it is possible to pose sophisticated queries to the underlying knowledge base. Additionally, with POOL it is possible to estimate the implication probability $P(d \rightarrow q)$ as a retrieval status value.

So why not just use POOL to model structured annotation hypertexts and perform annotation-based IR? POOL is a powerful framework, and one of our goals is to reuse its main ideas described above for our problem of annotation-based IR and for modelling structured annotation hypertexts. But as already outlined, annotation-based IR extends classical structured document retrieval which POOL mainly aims at. While POOL can indirectly model hypertexts and thus annotation hypertexts (by means of attributes and categories which can describe links), we want to represent and support some of the special elements of structured annotation hypertexts directly and less cumbersomely. POOL copes with tree structures, whereas

annotation hypertexts are not necessarily trees but directed graphs[4]. Furthermore, POOL neither supports fragments nor different kinds of annotations (like content and meta level ones). POOL's knowledge augmentation routines cannot take subcontexts which are established by annotations and referenced objects into account. We will later discuss another augmentation strategy (not supported by POOL) called *relevance augmentation*, which is useful in scenarios where we cannot access the knowledge in documents directly, but rely on the output of external retrieval services (such a situation is described in Agosti and Ferro (2005)). The bottom line is that representing structured annotation hypertexts in POOL would be at least cumbersome and would also lead to redundant information (e.g., when graphs need to be represented as trees) which possibly would be error-prone.

Nevertheless, most of the POOL concepts are very helpful for annotation-based retrieval, so one way to realise this and to model structured annotation hypertexts is to extend POOL accordingly. This also enables us to create a query language which is tailored to structured annotation hypertexts. This is exactly what the framework proposed in this thesis, POLAR, does. Below is a list of the features supported by POOL and POLAR, and those found in POLAR alone.

POOL features supported by POLAR:

- structured, complex objects (underlying tree structure)
- object-oriented modelling
- propositions: terms, attributes and categorisations
- four truth values (*true*, *false*, *inconsistent* and *unknown*)
- knowledge augmentation with subparts
- probabilistic inferencing and querying
- probabilistic retrieval ($P(d \rightarrow q)$)

Additional POLAR features not supported by POOL:

- structured annotation hypertexts (underlying graph structure)
- new annotation-based subcontexts (merged annotation targets, fragments, content and meta annotations, references)
- querying structured annotation hypertexts (structure queries)
- extended knowledge augmentation (considering peculiarities of subcontext types, e.g. negative annotations)
- relevance augmentation

POLAR can be compared to POOL in different ways. From an ontological point of view, POLAR is a specialisation of POOL since it inherits POOL's functionality, extends (e.g. by defining new kinds of access relations and subcontexts) and overrides it (by providing a different knowledge augmentation approach using annotations and referenced objects as additional

[4] In fact, as we will see later, the structure behind POLAR can even contain cycles, which could not be handled by POOL.

subcontexts besides subparts). From an implementation point of view (which is an issue in Chapter 6), POLAR is a kind of twin sister of POOL since both are based on four-valued probabilistic Datalog. From a functional point of view, POLAR extends POOL in that POOL programs can be evaluated by POLAR as well, but usually not the other way round.

4.2.2 Probabilistic Object-oriented Logics for Annotation-based Retrieval

POLAR has its roots in probability theory, object-oriented modelling and four-valued predicate logics.

Probability theory is a well-defined framework for capturing uncertain knowledge we are dealing with in IR. From this point of view, POLAR stands in the tradition of probabilistic retrieval models on the one hand and can be used as a vehicle to implement new probabilistic models on the other hand. Attributes, categorisations as well as index terms are assigned probabilities; from these probabilistic facts, new knowledge can be derived according to the rules of probability theory.

Object-oriented modelling is a well-known approach to create models of real world scenarios. In POLAR, documents and annotations are (complex) objects, which makes it compatible to the more general object-oriented view introduced in Chapter 3. On the global database level, objects can have attributes and they can be classified. POLAR supports the aggregation of objects by means of knowledge augmentation. Augmented contexts represent objects aggregated with their neighbouring context composed of annotations, referenced objects and the logical document structure. POOL's object-oriented features are preserved, but with an extended notion of annotation-based aggregation.

Four-valued predicate logics are used to model the content of complex objects as well as their relations to other objects. This content is represented as terms (propositions), classifications (unary predicates) and attributes (binary predicates) as well as access to other objects. Four-valued logics support the proper aggregation of objects by dealing with inconsistent knowledge coming from different sources.

The aim of the probabilistic, object-oriented logic-based framework is to support *annotation-based retrieval*. POLAR thus provides means to query structured annotation hypertexts on the one hand and methods based on probabilistic inference on the other hand.

In POLAR, we further distinguish between the object and the global database context. In the *object context*, classifications, attributes and terms allow for a sophisticated representation of objects, combining the content and logical view on documents together with annotations. Classifications and attributes represent factual knowledge about objects in the *global database context* (e.g., is an object a document or an annotation, and what kind of annotation). It also contains metadata about objects.

4.2.3 Document and Query Representation and Description

Recall the classes and properties of the object-oriented view on structured annotation hypertexts given in Figure 3.1 on page 24, and the conceptual retrieval model introduced in Section 4.1.2.1. In our further considerations, a document d in the conceptual model is an instance of AnnotatableObject in the object-oriented view. The transformation α_D turns the annotatable object d into its document representation $\underline{d}$. This representation has to be rich enough to capture the context of annotatable objects. It should not only contain the body of the object, for example as a bag of words in case of textual documents, but also its properties containing metadata and links to other objects, depending on the actual subclass of AnnotatableObject d

belongs to. For example, for instances of class Annotation, the object representation should contain information about the annotation target, other annotations annotating the object, the annotation type (meta or content level or any other subtype of these), which objects are referenced, scope, polarity, author, by which groups the annotation can be seen, links to subcomponents and fragments. We thus extract a very rich document representation $\underline{d}$ from the structured annotation hypertext which contains information about the direct context of d (objects directly connected to d) and which is transformed into a suitable document description d^D by the function β_D. d^D is a POLAR object, which we call a *context*. A context in the POLAR sense contains all information which describes an object, its content and its relation to other objects[5]. We will see examples of such contexts later.

A query q is transformed into a query representation $\underline{q}$ by α_Q. This representation is usually simpler than a document representation. For our purposes, it is enough to store the content of a query, for instance as a bag of words again. β_Q converts the query representation $\underline{q}$ into its description q^D. This query description is again a (simpler) POLAR context.

Besides documents and queries, some global knowledge might be extracted from the structured annotation hypertext, for example which groups a user is member of. Furthermore, it is possible to include external knowledge (e.g. coming from a thesaurus or ontology) into the POLAR knowledge base.

4.2.4 POLAR Knowledge Modelling

We continue with an informal introduction to POLAR which shows how structured annotation hypertexts can be modelled. We often refer to the object oriented model presented in the last chapter.

4.2.4.1 Classes and Is-A Relations

We start with a representation of the structured annotation hypertext on the class level (the T-Box). We create the following POLAR *rules*:

```
metaLevelAnnotation(O) :- highlighting(O)
metaLevelAnnotation(O) :- judgement(O)
contentLevelAnnotation(O) :- comment(O)
annotation(O) :- metaLevelAnnotation(O)
annotation(O) :- contentLevelAnnotation(O)
annotatableObject(O) :- annotation(O)
annotatableObject(O) :- document(O)
digitalObject(O) :- annotatableObject(O)
```

These rules express the "Is-A" relations in our model (capital letters denote variables) and further refine meta and content level annotations by introducing new subclasses Highlighting and Judgement of MetaLevelAnnotation, and Comment of ContentLevelAnnotation; every instance which is a highlighting or judgement is also a meta level annotation, every comment is a content level annotation, and so on. Note that we did not list the Component and Fragment classes here; these play a special role in POLAR as we will see later. By means of *categorisations* (or *classifications*), we assign instances to their corresponding classes; for example,

[5] "Context" in the POLAR sense is not to be mixed up with the notion of "context" in "context-based retrieval". In POLAR, a proposition is made within a certain document or annotation context, i.e. the proposition appears in the corresponding document or annotation. Each document or annotation is thus described by a "context" in the POLAR sense.

```
document(d1)
comment(a1)
```

means document(d1) and comment(a1), respectively. Due to the above rules, `annotatableObject(d1)`, `annotatableObject(a1)` and `annotation(a1)` is intensional knowledge derived implicitly.

4.2.4.2 Metadata

Metadata can be expressed by means of (possibly probabilistic) attributes and categorisations. For example,

```
d1.author(tim)
```

says that Tim is the author of d1.

The rules, categorisations and attributes presented above are all created in a certain context, the so-called *global database context*. But in POLAR, each document and annotation describes a context in its own right, as we are going to discuss now.

4.2.4.3 Documents and Annotations As Complex Objects

Complex Objects Documents and annotations in POLAR are complex objects and described by contexts containing probabilistic *propositions*, which can be terms, classifications and attributes. These propositions are derived from the document and annotation representation, which in turn is extracted from the structured annotation hypertext. Textual content is the source for *term propositions* in POLAR; their probability can be estimated using traditional *tf*-based measures normalised to the range between 0 and 1. Depending on the document type, there might also be multimedia content. Such content can be described with categorisations and attributes in POLAR. Furthermore, documents and even annotations might be structured. For example,

```
d1[ 0.5 information  0.6 retrieval
    0.7 digital  0.3 libraries
    s1[ 0.4 information  0.2 retrieval ]
m1[ o1[] o2[]
    house(o1) tree(o2) o2.leftOf(o1) ]
```

states that the context d1 can be described by the term propositions 'information', 'retrieval', 'digital' and 'libraries' with the corresponding probabilities (as the outcome of a text indexing process) and has a subpart (component) s1. s1 is a *subcontext* of d1. s1 can be indexed with 'information' and 'retrieval'. Furthermore, a multimedia object m1 might be described by a categorisation of its components and spatial properties, which can be expressed as *attributes*; in this example, it says that m1 has two components o1 and o2 which are a house and a tree, respectively, and o2 appears left of o1. With these mechanisms we are able to deal with structured textual and multimedia documents and annotations.

To extract the logical structure from the structured annotation hypertext and represent it in POLAR, we utilise the isPartOf relation: `d1[`p `s1[]]` iff isPartOf(s1,d1). p is the probability that we access s1 from d1, the so-called *access probability*. We come back to this probability in a later discussion. The propositions made in the context of a document or annotation and their weights are derived by indexing the body of the corresponding DigitalObject instance.

Four Truth Values As mentioned before, annotations can contain contradictions and thus inconsistent knowledge. To represent documents and annotations, POLAR therefore utilises four-valued logics. We do not only cope with the classical truth values *true* and *false* for propositions, but also introduce two additional truth values, namely *inconsistent* and *unknown* (Belnap, 1977). POLAR implicitly deals with an open world assumption. This means that if there is no evidence that a proposition is *true* (*false*), this does not imply that it is *false* (*true*). In the probabilistic case, if we assume a proposition to be *true* with a probability of p, we cannot assume that it is *false* with a probability of $(1-p)$. In POLAR, we can give probabilities for the four truth values directly, with one constraint: the sum of these probabilities must be 1. Furthermore, we regard the cases that a proposition is *true*, *false*, *inconsistent* or *unknown* as disjoint events.

Negative and Positive Evidence Based on the notion of the four truth values we introduce *positive* and *negative evidence*. A proposition is positive if it is *true* or *inconsistent*, and it is negative if it *false* or *inconsistent*. This is explained by the view of the four truth values as sets. In particular, $true = \{t\}, false = \{f\}, inconsistent = \{t, f\}$ and finally $unknown = \emptyset$. A proposition is positive if its truth value contains 't' in its set notation, and it is negative if it contains 'f' in its set notation. Inconsistent knowledge is both positive and negative; the truth value *true* means *positive and not negative*, whereas *false* means *negative and not positive*.

Example 2 (Four-valued knowledge modelling): An application of modelling with four truth values is to reflect cases where we know with a certain probability that a document is about a term or about its negation. For example

```
d1[ 0.8/0.1 ir    0/0.4 db ]
```

says that 'ir' is *true* in d1 with 0.8 probability and false with a probability of 0.1. An even more extreme case is the term 'db', from which we only know that it is *false* with 0.4 probability. Since for the term 'ir', the probabilities do not amount to 1, we implicitly assume a probability of $1-0.8-0.1 = 0.1$ that 'ir' is *unknown* in d1. Similar for the term 'db' where the probability that it is *unknown* in d1 is $1-0.4 = 0.6$. An alternative interpretation of the truth value *false* is that instead of saying that a document is about the negation of a term, it is explicitly *not* about the concept the term describes.

Not only the probabilities of *true* and *false* can be given directly, but also a value for *inconsistent*.

```
d2[ 0.3/0.2/0.4 libraries ]
```

says that libraries is *true* in d2 with 0.3 probability, *false* with 0.2 probability and *inconsistent* with a probability of 0.4. Although it would be syntactically correct, we do not need to give a probability for *unknown* because this value is derived from the others ($1-(0.3+0.2+0.4) = 0.1$ is then the probability that libraries is *unknown*). □

There are shortcuts for the non-probabilistic case. For example, `d3[ ir ]` is equal to `d3[ 1 ir ]` and `d3[ 1/0 ir ]`. Additionally, `d4[ !db ]` equals `d4[ 0/1 db ]`.

4.2.4.4 Structured Annotation Hypertexts and Threads in POLAR

We have discussed how the content and logical structure of a complex document or annotation can be described in POLAR by means of contexts, subcontexts, propositions and four truth

values. This is the part which POLAR inherits from POOL. Now we have to connect documents, fragments and annotations, which are special kinds of subcontexts, according to the given structured annotation hypertext; this is not supported by POOL.

Content Level Annotations `d1[`p `*a1]` means that the content annotation a1 annotates d1. Formally, `d1[`p `*a1]` iff hasAnnotationTarget(a1,d1) $\wedge$ ContentLevelAnnotation(a1). p denotes the access probability again. `d1[0.8 *a1]`, for example, means that document d1 is annotated by a1 and this annotation is accessed (or *considered*) with 0.8 probability.

Meta Level Annotations Meta annotations make assertions about objects on the meta level. A judgement j1 about a document d1 saying "this is a good introduction" might be indexed and modelled in POLAR as

```
d1[ 0.5 information  0.6 retrieval
    0.7 digital  0.3 libraries
    0.7 @j1 ]
j1[ 0.3 good  0.7 introduction ]
```

In general, `d1[`p `@j1]` means that there exists an annotation j1 which makes assertions about d1 on the meta level and is accessed from d1 with probability p. More formally, `d1[`p `@j1]` iff hasAnnotationTarget(j1,d1) $\wedge$ MetaLevelAnnotation(j1).

Polarity Another attribute of annotations we identified before is their polarity. The polarity of an annotation might be explicitly determined by, e.g., the annotation type or just be modelled in the polarity attribute of an Annotation object in the structured annotation hypertext. When an annotation type or a polarity is not explicitly given, machine learning algorithms could be applied to determine the polarity (Lechtenfeld, 2007), similar to sentiment classification (Pang et al., 2002). If we imply that the examples above model positive content or meta annotations, we have to provide expressions for negative annotations. `d1[`p `-*a1]` (`d1[`p `-@a1]`) says that a1 is a negative content (meta) annotation of d1 with the respective access probability p.

References *References*, which are also a component of annotation hypertexts, are syntactically represented in POLAR as `a1[=>o1]`, which means object o1 is referenced by annotation a1. Formally, `a1[`p `=>o1]` iff references(a1,o1). p is the probability that we access the referenced object o1 from a1.

Due to references, an annotation can link objects in the repository. For example,

```
d1[ 0.7 *a1 ]
a1[ 0.8 =>d2 ]
d2[]
```

creates a link between d1 and d2 through the annotation a1. The probability that we access d2 from a2 is 0.8. The probability that d2 is accessed from d1 is $0.7 \cdot 0.8 = 0.56$. a1 thus realises a link between d1 and d2 which is not provided by the author of d1. As a side effect of introducing references in our framework, we are also able to represent links given by the author of a document; the expression `d1[=>d2]`, for instance, would model a link between d1 and d2 provided by the author of d1. This makes POLAR a tool for representing hypertexts in general (without annotations), although this is not the main focus of POLAR.

Fragments When users create an annotation about a certain passage of a document, they first select the corresponding document fragment. This fragment is also a part of a document, and the fact that this was an annotation target should be expressed in our framework as well, since this knowledge can be valuable in the retrieval process. For example,

```
d1[ 0.5 information  0.6 retrieval
    0.7 digital  0.3 libraries
    0.8 f1|| 0.9 digital  0.5 libraries  0.7 *a1|| ]
```

means that a fragment f1 of d1 which is about digital libraries was selected as an annotation target for a1; we refer to this fragment as an *annotated part* of d1. Formally, `d1[`p_1 `f1||` p_2 `*a1||]` iff isFragmentOf(f1,d1) and hasAnnotationTarget(a1,f1). p_1 is the probability that we access the fragment f1 from d1, and p_2 is the probability that we access a1 from f1.

Fragments have a special property regarding annotations. We say that if an annotation annotates a fragment, it also annotates the object the fragment belongs to. For example, if a user selects a part of a paragraph and annotates this fragment, we regard the annotation as belonging to both the fragment and the paragraph. Therefore, `d[f||` p `*a||]` implies `d[` p `*a]`. We call this special property *fragment permeability*.

Merged Annotation Targets Annotation targets (i.e. the objects or fragments which are annotated) may contain important information to determine the relevance of an annotation. As a simple intuitive example, consider a fragment about digital libraries which is annotated with a comment "This is an important new technology". A reader of this annotation has to refer to the annotation target to resolve the anaphora "this" and to learn that the annotation talks about digital libraries. We see that the content of annotation targets is an important context when searching for annotations, which is also confirmed later by the experiments in Chapter 8.

Consider the following example:

```
a1[ 0.8 t1< 0.7 digital  0.8 libraries >
    0.6 important  0.8 new  0.7 technology ]
```

t1 is the *merged annotation target* (or shortly *merged target*) of a1 and is about digital libraries. More generally we say that an expression `a1[`p `t1<...>]` states that t1 is the merged target of a1 and that this context is accessed with probability p. Merged annotation targets are constructed as follows. Let $T_a = \{o|\mathsf{hasAnnotationTarget}(a, o)\}$ denote the set of annotation targets of a. Instead of considering each annotation target on its own, we see content an annotation refers to in an integrated way and create a new virtual document t which contains all propositions of each of a's annotation targets. Therefore, $t = \cup_{o \in T_a} o$. In the indexing step, probabilities are calculated for each proposition, e.g. based on the term frequency and length of the newly created document t in case of terms. One reason to do so comes from our view of emails as annotations, as it is discussed later in Section 7.1.2.

4.2.4.5 Special Commands

POLAR supports a set of special commands, which are prefixed by a "_". For example,

```
_echo("Print me")
```

prints the string "Print me" to the console. We will introduce additional special commands when appropriate.

4.2.4.6 POLAR Symbols and their Associations

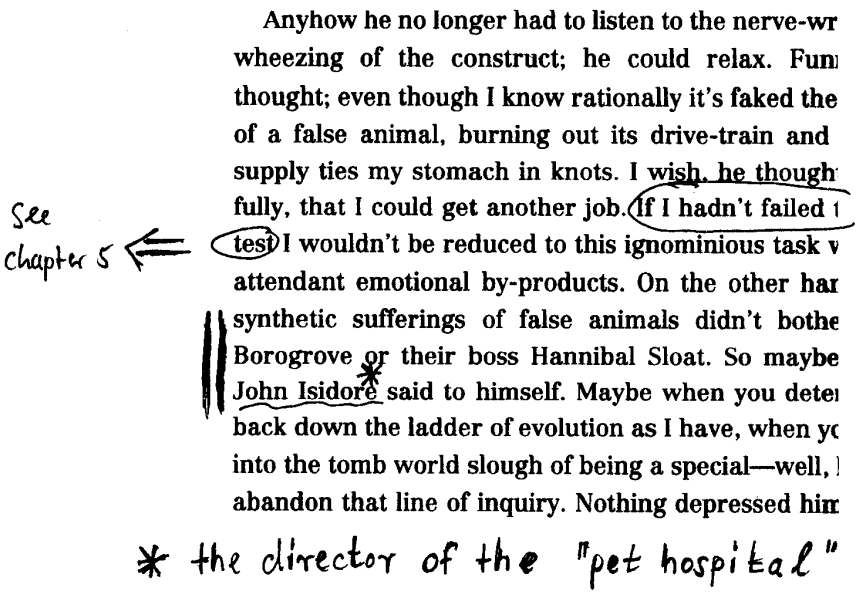


Figure 4.2: An annotated snippet

The syntactical elements which denote subcontexts in POLAR are chosen with certain associations in mind. Subpart access, denoted as "`s[...]`", follows the notation already known from POOL. Regarding the symbols for annotations and references, consider the annotated snippet from Philip K. Dick's novel "Do Androids Dream of Electric Sheep?" shown in Fig. 4.2. The annotations follow a certain annotation code, which differs from person to person. In this snippet, a "*" is used to write a comment about "John Isidore". The corresponding POLAR element is a content annotation "`*a1`" (if we give this annotation the ID "a1"). The fragment starting with "synthetic sufferings" is marked by two vertical bars highlighting this fragment. The POLAR variant would be something like "`f1||synthetic suffering ...||`". Furthermore, the annotation belonging to the fragment "If I hadn't failed...test" is a reference (denoted by the "⇒" symbol) to another structural element of the novel, the chapter 5. In POLAR, this could be represented as "`a2[=>chapter5]`". Meta level annotations are distinguished from content annotations with the "`@`" symbol, saying that the annotation says something "@bout" the annotated object. Finally, from emails we know that their quoted part is often identified by a ">" character preceding each line of the quotation, which motivated the usage of "<" and ">" to denote merged annotation targets (see also the representation of emails in POLAR later in Section 7.1.2).

4.2.5 Querying and Retrieval in POLAR

While the creation of the knowledge base described above is the outcome of a context-based indexing step, we are going to discuss possible querying and retrieval options in POLAR now. *Queries* are expressed as headless rules, with variables in capital letters. When queries are evaluated, these variables are substituted by the constants (object ids) appearing in a POLAR program, and their probabilities are used to create a ranking.

4.2.5.1 Database Queries

Database queries return facts from the given knowledge base in the global database context. Suppose that annotations and documents have a property author denoting the author of an annotation. The query

```
?- A.author(turner) & annotation(A)
```

returns Turner's annotations. Similarly,

```
?- A.author(turner) & document(A)
```

returns all of Turner's documents.

```
?- d1.author(A)
```

yields all authors of d1.

4.2.5.2 Structure Queries

POLAR offers means to pose structure queries to the structured annotation hypertext. Results are ranked based on the corresponding access probabilities.

```
?- d1[*A]
```

returns all (positive) content annotations annotating d1, whereas

```
?- D[*a1]
```

returns all objects annotated by the content annotation a1. Similarly,

```
?- d1[@A]
```

yields all positive meta annotations of d1, whereas

```
?- d1[-@A]
```

fetches all its negative meta annotations.

```
?- d1[||F]
```

yields all fragments of d1. For references,

```
?- a1[=>O]
```

returns all objects referenced by a1 and

```
?- A[=>d1]
```

yields all objects referencing d1. An example of a more complex structure query is

```
?- d1[*A] & A[=>O]
```

which returns pairs of objects (a, o) where a is a content annotation of d1 and o is an object a refers to. This query fetches all objects which are linked from d1 through an annotation together with the corresponding annotation. If we are only interested in the linked objects and do not care through which annotation this link was established, we need to apply rules:

```
linked_from_d1(O) :- d1[*A] & A[=>O]
?- linked_from_d1(O)
```

The rule sorts out any annotation and only preserves the objects they link to (which are then fetched by the query).

4.2.5.3 Content-oriented Queries

Content-oriented queries deal with uncertain knowledge and calculate a value for each object w.r.t. the query and according to the probabilities of their propositions.

```
?- A[ search ]
```

returns all objects containing 'search'.

```
?- D[ information & retrieval ]
```

returns all documents about "information AND retrieval".

POLAR provides a special syntax to query for fragments explicitly. The query

```
?- F|| digital & libraries ||
```

returns annotated parts about digital libraries. This kind of query enables direct access to annotated document fragments in case users are only interested in these parts.

4.2.5.4 Retrieval by Implication Probability

The content queries so far exploit the content of objects to fetch them if they fit to the given POLAR query. If, for example, the probability of term propositions is based on the within-term frequency (*tf*) of objects, we gain a *tf*-like ranking. But information retrieval approaches usually also employ the inverse document frequency *idf* to calculate a RSV. Furthermore, we want to support retrieval based on the probability $P(d \rightarrow q)$ that a document d implies a query q is computed. For this we define a query[6] q as another context. For example,

```
q1[ information   0.8 retrieval ]
```

defines a query q1 containing the terms 'information' and 'retrieval'. 'retrieval' is weighted with a probability of 0.8. The POLAR query

```
?- D->q1
```

returns all objects which imply this query, ranked by their decreasing implication probability. How this value is actually calculated can differ. For example, Wong and Yao (1995) show how probabilistic inference can be interpreted to realise well-known retrieval functions like the vector space model. Rölleke (1998) presents how this interpretation can be applied to POOL. In principle, it is possible to assimilate this solution for POLAR as well. In Section 6.2.4 we will discuss further retrieval functions based on the implication probability, which are able to produce a $tf \times idf$-like ranking. In the remainder of this chapter, in particular in Section 4.2.6.2, we apply a very simple estimation for $P(d \rightarrow q)$, which takes for each query term the product of the term's *tf* within d and q, and its *idf* value.

In order to allow for the integration of *idf*-like values, POLAR introduces *term spaces*. For example,

```
0.5 °retrieval
```

says that the probability of the term 'search' (which can be based on the inverse document frequency, depending on the application) is 0.5.

[6] "Query" is meant in a retrieval sense here, not to be mixed up with a POLAR query prefixed by "`?-`"

Example 3 (Retrieval by implication probability): Consider the following knowledge base and query:

```
0.7 °information
0.5 °retrieval
d1[ 0.5 information  0.6 retrieval ]
q1[ information   0.8 retrieval ]
```

The query

```
?- D->q
```

returns d1 with an RSV of 0.506. This is calculated as

$$\overbrace{1}^{\texttt{q1[information]}} \cdot \overbrace{0.5}^{\texttt{d1[information]}} \cdot \overbrace{0.7}^{\texttt{°information}} + \overbrace{0.8}^{\texttt{q1[retrieval]}} \cdot \overbrace{0.6}^{\texttt{d1[retrieval]}} \cdot \overbrace{0.5}^{\texttt{°retrieval}} -$$
$$\overbrace{1}^{\texttt{q1[information]}} \cdot \overbrace{0.5}^{\texttt{d1[information]}} \cdot \overbrace{0.7}^{\texttt{°information}} \cdot \overbrace{0.8}^{\texttt{q1[retrieval]}} \cdot \overbrace{0.6}^{\texttt{d1[retrieval]}} \cdot \overbrace{0.5}^{\texttt{°retrieval}}$$
$$= 0.506$$

according to the inclusion-exclusion formula (each occurrence of a proposition in a context is regarded as an event) which we formally introduce later on page 133. □

4.2.6 Knowledge and Relevance Augmentation

The modelling, querying and retrieval facilities introduced so far see each context individually. Even subparts in complex objects are viewed separately from the object they are contained in, or, in other words, a complex object does not "know" anything about its subparts, annotations, fragments and targets. Our goal is to use these subcontexts to answer queries about complex objects or for retrieval. The knowledge contained in a context and connected subcontexts should be aggregated. This is done by means of *augmentation*. POLAR supports two augmentation strategies: knowledge augmentation and relevance augmentation.

4.2.6.1 Augmented Contexts

The reason why we did not just model properties like $\mathsf{hasAnnotationTarget}(a, d)$ as a POLAR property `a.hasAnnotationTarget(d)` becomes clear when introducing the concept of augmented contexts. As said before, in POLAR every object establishes a *context*. Within their context, objects have a specific knowledge determined by their content. Consider the following example:

```
d1[ 0.5 information  0.6 retrieval
    0.7 digital  0.3 libraries
    0.8 *a1 ]
a1[ 0.6 search  0.8 big  0.7 issue ]
```

a1 and d1 establish a respective context. d1 knows about 'information', 'retrieval', 'digital' and 'libraries' in its context, but nothing more, whereas a1 knows about 'search', 'big' and 'issue'. Our representation also states that content annotation a1 is a *subcontext* of d1 which is accessed from d1 with probability 0.8. Also subparts, annotated fragments and merged annotation

targets are considered as subcontexts. If we access a1 from d1 (or in general a subcontext from its supercontext), we create an *augmented context* d1(a1). Augmented contexts aggregate the information in all contained contexts. For example, knowledge augmentation propagates all propositions in the subcontexts to the augmented context, according to their respective access probabilities. In our example above, 'retrieval' is known in d1 with 0.6 probability, while 'search' is completely unknown in this context. But if we access a1 from d1, which we do with 0.8 probability, there is a further probability of 0.6 that the context a1 knows about 'search'. The probability that the augmented context d1(a1) knows about 'search' is determined by the probability that d1 knows about search or we access a1 from d1 and a1 knows about 'search'. Therefore, d1(a1) knows about 'search' with a probability of $0.8 \cdot 0.6 = 0.48$. Relevance augmentation, on the other hand, propagates retrieval status values. For both kinds of augmentation, we have to care about some peculiarities of the different context types. For example, meta annotations are ignored as subcontexts for augmentation, since otherwise we would mix information on the content and the meta level.

Before we discuss knowledge and relevance augmentation any further, we define the notion of augmented context expressions.

Definition 3 (Augmented context expression):
We call expressions like d1(a1), where we denote that d1 is the context to augment and a1 is a (not further specified) subcontext, an *augmented context expression*. Augmented context expressions can be nested, so that d1(a1(a2)) means that a1 is a subcontext of d1 and a2 a subcontext of a1. The subcontext relation is transitive, so a2 is a subcontext of d1 as well. Here, a1 is the *direct subcontext* of d1. Contexts can have more than one direct subcontext. In the augmented context expression, these are separated by commas. For instance, d1(a1,a2(a3,a4)) means that both a1 and a2 are direct subcontexts of d1, and a3 and a4 are direct subcontexts of a2.

We now present the two augmentation strategies, knowledge and relevance augmentation. We start with knowledge augmentation and present examples in order to illustrate the approach. Further discussions of both augmentation approaches can be found in the two subsequent chapters.

4.2.6.2 Knowledge Augmentation

Some examples shall discuss the effect of knowledge augmentation w.r.t. the certain POLAR subcontext types and how results are computed.

Fragments Recall the previous example:

```
d1[ 0.5 information  0.6 retrieval
    0.7 digital  0.3 libraries
    0.8 f1|| 0.9 digital  0.5 libraries  0.7 *a1|| ]
d2[ 0.5 libraries ]
a1[]
```

A query for documents about 'libraries' without knowledge augmentation yields:

```
?- D[ libraries ]
0.5 (d2)
0.3 (d1)
```

due to the weight of 'libraries' in d1 and d2 which represents the probability that this term is *true* in d1 and d2. Now we pose the same query, but with knowledge augmentation:

```
?- //D[ libraries ]
0.58 (d1)     # from d1(a1,f1)
0.5  (d2)     # from d2
```

The "//" tells the system to perform knowledge augmentation. What happened here? The knowledge of d1 is augmented with the knowledge we find in its fragment f1. The fact that this fragment has been annotated (otherwise the fragment would not exist) makes this fragment an important part of d1, since the annotator spent some time to annotate it. Our claim is that such annotated passages are *implicitly highlighted* (this is especially true for fragments which are explicitly highlighted, e.g., by marking or underlining them). The more users annotate a specific passage (implicitly or explicitly), the more we get an *n-way-consensus* (Marshall, 1998) that this passage has some value in it. The hypothesis is that we thus receive additional evidence that the corresponding document should be indexed with the propositions (terms) in the annotated part, resulting in higher probability of these propositions in d1(f1). So the effect of knowledge augmentation with fragments is that no new terms are introduced, but the weights of existing ones in the augmented context are raised, causing a different ranked result in the example above. The new weight for 'libraries' represents the probability that the term is *true* in the augmented context d1(a1,f1), which is the sum of the probabilities of the following four cases:

- 'libraries' is *true* in d1 and we do not access/consider f1 (probability $0.3 \cdot (1-0.8) = 0.06$);
- 'libraries' is *true* in d1 and we access f1 and the term is *unknown* in f1 ($0.3 \cdot 0.8 \cdot 0.5 = 0.12$);
- 'libraries' is *true* in d1 and we access f1 and the term is *true* in f1 ($0.3 \cdot 0.8 \cdot 0.5 = 0.12$);
- 'libraries' is *unknown* in d1 and we access f1 and the term is *true* in f1 ($0.7 \cdot 0.8 \cdot 0.5 = 0.28$);

The sum of the probabilities of these disjoint events is $0.06 + 0.12 + 0.12 + 0.28 = 0.58$. It represents the probability that an event occurs which makes 'libraries' *true* in d1(f1). This probability can alternatively be calculated as

$$P(\overbrace{\text{'libraries' } \textit{true} \text{ in d1}}^{=0.3} \text{ OR } \overbrace{\text{f1 accessed from d1 and 'libraries' } \textit{true} \text{ in f1}}^{=0.8 \cdot 0.5 = 0.4}) = 0.3 + 0.4 - 0.3 \cdot 0.4 = 0.58$$

with the inclusion-exclusion formula (see Definition 24 on page 133).

Note that we use the augmented context d1(a1,f1) instead of d1(f1(a1)), as the structure indicates it. This has two reasons: first, due to fragment permeability, if a1 annotates f1 and f1 is a fragment of d1, a1 also annotates d1. Second, since we expand d1 with a1 directly, we do not consider a1 any more when accessing it from f1.

Merged Targets In contrast to fragments, merged targets usually add new terms and propositions to augmented contexts. This is a very important feature of merged targets, since our motivation to introduce them was to gain additional information about the topics an annotation is about. Consider again the example from page 59:

```
annotation(a1)
a1[ 0.8 t1< 0.7 digital  0.8 libraries >
    0.6 important  0.8 new  0.7 technology ]
```

A query for annotations about "digital AND libraries", in POLAR expressed as

```
?- A[ digital & libraries ] & annotation(A)
```

would not return a1, since these terms are not both *true* in a1. Nevertheless, a1 talks about digital libraries when it states that it is an "important new technology". So a1 is relevant to the query. Since the crucial information that a1 is relevant to queries for digital libraries is contained in the merged target, we apply knowledge augmentation and get

```
?- //A[ digital & libraries ] & annotation(A)
0.3584 (a1)     # from a1(t1)
```

The probability that 'digital' is *true* in the augmented context a1(t1) is $0.8 \cdot 0.7 = 0.56$; for 'libraries', this is $0.8 \cdot 0.8 = 0.64$. The resulting probability is thus $0.56 \cdot 0.64 = 0.3584$.

Subparts Knowledge augmentation with subparts is the traditional application of the POOL framework (Rölleke, 1998) which sees a structured document as an aggregation of its subparts. Since POLAR extends POOL, this view is applied in POLAR as well and we also augment contexts with their subparts. This allows for finding a best entry point within the document structure. Consider the example

```
d1[ 0.7 s1[ 0.8 ir ]
    0.7 s2[ 0.9 db ] ]
```

A query

```
?- D[ ir & db ]
```

returns nothing, since in none of the contexts, both query terms are *true*. The query

```
?- //D[ ir & db ]
```

returns

```
0.3528 (d1)     # from d1(s1,s2)
```

since in the augmented context d1(s1,s2), 'ir' is *true* with a probability of $0.7 \cdot 0.8 = 0.56$ and 'db' with $0.7 \cdot 0.9 = 0.63$ probability, leading to an overall probability of $0.56 \cdot 0.63 = 0.3528$ for d1. Each of the subparts s1 and s2 does not satisfy the query, but the augmented context d1(s1,s2) does, since it aggregates both s1 and s2. The query

```
?- //D[ ir ]
```

yields

```
0.8  (s1)     # from s1
0.56 (d1)     # from d1(s1,s2)
```

which is the desired result, since s1 and (the augmented) d1 both deal with 'ir', but s1 is more specific and a user would not miss anything if she only considers s1 and not the whole document d1.

Positive Content Annotations With fragments and merged targets, we considered objects created during the annotation process for knowledge augmentation. A natural step further is to take the content of annotations into account as well.

Consider the following POLAR program (imagine for example a document about 'soccer' and an annotation saying that 'soccer' is called 'football' in Europe):

```
document(d1)
annotation(a1)
d1 [ 0.6 soccer
     0.7 *a1 ]
a1 [ 0.5 football ]
```

The query for documents about football

```
?- D[ football ] & document(D)
```

would not retrieve d1, although (for Europeans) it would be relevant. The query

```
?- //D[ football ] & document(D)
```

considers the term 'football' in a1 and would thus retrieve d1 with a probability of $0.7 \cdot 0.5 = 0.35$ due to the association of d1 with 'football' in d1(a1).

An interesting application of knowledge augmentation is the handling of *contradictions*, which often occur in annotations and especially discussions. Consider the following situation:

```
d1[ *a1  *a2 ]
a1[ moon_made_of_cheese ]
a2[ !moon_made_of_cheese ]
```

Annotation a1 states that the moon is made of cheese, and a2 says it is not. A reader of d1 would not get any information about what the moon is made of at all; but if she considers the annotations as well, she would get inconsistent information about the moon being made of cheese; neither the query "`?- //D[moon_made_of_cheese]`" nor "`?- //D[!moon_made_of_cheese]`" would return d1, because 'moon_made_of_cheese' is *inconsistent* in d1(a1,a2)[7].

We extend the example above with probabilities:

```
d1[ *a1  *a2 ]
a1[ 0.8 moon_made_of_cheese ]
a2[ 0/0.7 moon_made_of_cheese ]
```

The query

```
?- //D[ moon_made_of_cheese ]
```

returns

```
0.8  (a1)     # from a1
0.24 (d1)     # from d1(a1,a2)
```

We can see here how the negative probability (0.7) in a2 influences the probability that 'moon_made_of_cheese' is *true* in d1(a1,a2). This value is calculated as $0.8 \cdot (1 - 0.7) = 0.24$. The query "`?- //D[!moon_made_of_cheese]`" returns d1 with a probability of $(1 - 0.8) \cdot 0.7 = 0.14$.

[7] Note that POLAR does not offer means to query inconsistent knowledge yet. A possible extension of POLAR might evaluate the query "`?- //D[moon_made_of_cheese & !moon_made_of_cheese]`" in a way that it returns d1 in our example. This might be interesting for tasks where one wants to explicitly search for topics which are discussed controversially.

Negative Content Annotations The examples presented so far all dealt with positive annotations. The question is how we can handle negative annotations w.r.t. knowledge augmentation. One option is that a proposition `a` appearing in a negative content annotation is propagated as ¬`a` in the augmented context.

Consider the example of an annotation thread in COLLATE, which is shown in Figure 2.1 on page 13. We see the annotation, an interpretation saying that the film mentioned in the source document was censored for political reasons. This annotation is annotated again; in the reply a2, the annotator expresses her disagreement with the previous statement by using a *counterargument* annotation type and saying that she thinks there were no political reasons ("I disagree. There were no political reasons").

We want to model this situation in POLAR and also reflect the fact that a2 is a negative response to a1, especially regarding the topic "political reasons". One option is to detect in a2's content that the "no" belongs to "political reasons". We might then assign "`!political_reasons`" to a2. The other option (for instance in case of an automatic indexer which is based on terms and does not detect that "no" belongs to "political reasons" or even treats "no" as a stop word) is to use negative polarity and create a negative content annotation, if we assume this can be inferred from the annotation type (counterargument in this case). This scenario is expressed in POLAR as

```
document(d)
annotation(A) :- interpretation(A)
annotation(A) :- counterargument(A)
interpretation(a1)
counterargument(a2)
 d[ *a1 ]
a1[ 0.7 film   0.5 censored   0.8 political_reasons   0.8 -*a2 ]
a2[ 0.7 political_reasons ]
```

The first line says that d is a document. The second and third line mean that every interpretation or counterargument is an annotation. Line 4 and 5 classify a1 and a2 as interpretation and counterargument, respectively, which also means they are annotations. Line 6 introduces document d, with no further (textual) content. Line 7 shows the annotation a1 and its corresponding terms and term weights. In the context of a1, a2 is a negative content annotation and is accessed with 0.8 probability. The last line shows a2. The query

```
?- D[ political_reasons ]
```

returns, without any augmentation,

```
0.8 (a1)
0.7 (a2)
```

The document d would not be retrieved.

We have two annotations a1 and a2 which are both about 'political reasons', but a2 talks negatively about a1 with respect to this term and is thus a negative content annotation in the context of a1. We interpret this situation that a2 attacks the fact that a1 is a good authority for 'political reasons', which means that the corresponding term weight should be decreased when considering the augmented context a1(a2). Knowledge augmentation adds the probability that 'political reasons' is *true* in a2 to the probability that it is negative in a1(a2); this value is then also propagated to d1(a1(a2)). So we get

```
?- //D[ political_reasons ]
0.7  (a2)     # from a2
0.24 (a1)     # from a1(a2)
0.24 (d)      # from d1(a1(a2))
```

While a1 has a positive effect on d(a1), We see that the existence of a2 has a negative effect on a1(a2) and therefore also d(a1(a2)). Without a2, a1 and d would be assigned a value of 0.8 instead of 0.24 ($= 0.8 \cdot (1 - 0.7)$, the probability that 'political reasons' is positive and not negative in a1(a2) and d(a1(a2)), respectively). If there was another negative content annotation a3 which annotated a2 and also contains the term 'political reasons', then a2 would have a negative effect on a1(a2(a3)) and d(a1(a2(a3))), but a3 would in turn have a positive effect on a1(a2(a3)) and d(a1(a2(a3))) since it has a negative effect on a2.

Implication Probability and Knowledge Augmentation As mentioned before, POLAR supports context implication for retrieval, i.e. it estimates the probability $P(d \rightarrow q)$ that a document implies a query. This can be combined with knowledge augmentation to realise annotation-based retrieval.

Again, we calculate the implication probability as above in Section 4.2.5.4. Besides the weights of propositions (terms, attributes and classifications) in contexts, this method to compute the implication probability also integrates the inverse document frequency. Let us extend the above COLLATE example with some *idf*-based term measures, for instance

```
0.5 °political_reasons
```

which says that the (*idf*-based) term probability of 'political reasons' is 0.5. Then,

```
q[ political_reasons ]
?- D->q
```

calculates a retrieval status value of 0.4 for a1 (the term weight (0.8) multiplied with the *idf* (0.5)) and 0.35 for a2. d1 would not be found in that case. Now we apply knowledge augmentation. We get

```
?- //D->q
0.35 (a2)     # from a2
0.12 (a1)     # from a1(a2)
0.12 (d)      # from d(a1(a2))
```

As seen above, the weight for 'political reasons' in the augmented contexts of a1 and d is 0.24, and $0.24 \cdot 0.5$ is the RSV of both a1 and d.

We introduced POLAR's knowledge augmentation facilities. These take into account every subcontext of the context to augment. Later in Section 4.3.1 we discuss further knowledge augmentation examples and how to fine-tune the augmentation process according to subcontext types.

4.2.6.3 Relevance Augmentation

Besides knowledge augmentation, POLAR supports another augmentation strategy which we call *relevance augmentation*. In contrast to knowledge augmentation, we calculate the RSV of

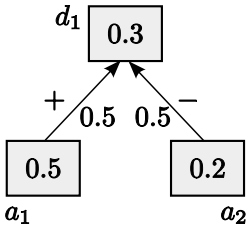


Figure 4.3: Relevance augmentation example. Arrows denote propagation.

each single object first. Relevance augmentation then means that we propagate the retrieval status value of a context to its supercontext to create the final RSV of the augmented context (see the example below). The advantage of relevance augmentation is that it operates on retrieval status values rather than propositions. This is important for example in cases when an annotation service employing POLAR does not have access to document full texts, which are possibly stored in external repositories, to extract the required propositions for knowledge augmentation. Such a scenario is outlined in Agosti et al. (2006) where the DiLAS annotation service is presented; DiLAS is supposed to be linked to several external digital library management systems, and these systems might only provide document handles and a search API, but no access to the full texts in order to index them. With relevance augmentation, a POLAR implementation can query these external sources in order to fetch retrieval status values for external documents, and merge them with annotations' retrieval status values for annotation-based document search (as outlined similarly in Agosti and Ferro (2005))[8].

Relevance augmentation is illustrated in Figure 4.3. Let us say that for a query q1 and a document d1, an external digital library management system returns a retrieval status value of 0.3. Furthermore, the RSVs for a1 and a2 are 0.5 and 0.2, respectively. Let us further assume that a1 is a content annotation and a2 is a negative content annotation of d1. The corresponding access probabilities are 0.5. The relevance augmentation approach now combines these three retrieval weights and generates a new context-based one for d1. The weight of a1 raises the resulting weight for d1, while the weight of a2 lowers it again, since we have negative evidence about the relevance of d1 here. From the document itself, we know that it is relevant with 0.3 probability. From the annotations, we have both positive and negative evidence. The positive evidence comes from a1; together with the positive evidence from the document, we infer with a probability of $0.3 + 0.5 \cdot 0.5 - 0.3 \cdot 0.5 \cdot 0.5 = 0.475$ that d1 is relevant. Considering the negative evidence from the annotations, we infer that d1 is not relevant with $0.5 \cdot 0.2 = 0.1$ probability. Relevance augmentation combines positive and negative evidence and calculates the probability that we have positive evidence and not negative evidence from the context, that is $0.475 \cdot (1 - 0.1) = 0.4275$, which is the final context-based retrieval status value of d1.

Syntactically, the expression "`?- //D->q1`" could be used for relevance augmentation[9]. See Section 6.2.5 for a further discussion.

[8] We have to be aware that the RSVs coming from external sources are not necessarily probabilities and often need to be normalised accordingly.

[9] Note that, despite of the choice of the syntactic expression, relevance augmentation does not necessarily calculate an implication probability (this also depends on the external sources and their underlying retrieval function).

We introduced POLAR's knowledge modelling, querying and retrieval capabilities and its core concept, augmentation. Before discussing POLAR's syntax and semantics formally in the next chapter, we present some further examples of possible POLAR applications.

4.3 Further Application Showcases

The structured annotation hypertext introduced in the last chapter is a very complex data structure containing many different components and their relations. As we have seen in Chapter 2, annotations can be of many different types and can have many facets. This makes clear that POLAR, as a framework for modelling structured annotation hypertexts, to query them and to perform probabilistic retrieval on them, potentially serves a wide range of possible tasks and applications. In the previous considerations, we have already seen some examples, when we discussed knowledge augmentation, which is able to handle negative and inconsistent knowledge. We are now going to present some further application showcases, also to give additional examples of POLAR programs. These single examples can of course be combined in order to fulfil more complex tasks. All showcases have in common that they combine different kinds of evidence coming from the structured annotation hypertext in order to determine the relevance of documents and annotations, respectively, for document or discussion search.

4.3.1 Annotation-based Structured Document Retrieval and Discussion Search

4.3.1.1 Outline

We have discussed POLAR's main retrieval function based on the estimation of $P(d \rightarrow q)$, which can be used in combination with knowledge and relevance augmentation. Augmentation is a well-known principle for structured document retrieval, where we search for a best entry point within a structured document. From this perspective, augmentation in POLAR allows for *annotation-based structured document retrieval*. Annotations help to find best entry points in documents. Consider the situation illustrated in Figure 4.4. When we perform classical structured document retrieval, we are interested in documents and their subparts, which are in this case d1, s1,s2, s11, s12 and s21. Propagation and augmentation considers these subcontexts only. For instance, just s21 would influence the RSV of s2 (by propagating its knowledge to s2 when forming s2(s21)). If we add annotations to the retrieval process (by taking a1, ..., a7 into account as well), s2's RSV is also influenced by its direct annotations a3 and a6, and further indirectly by a4 and a5. It is also indirectly influenced by a7, since the knowledge of a7 is propagated to s21 and (with a lower resulting propagation factor) also to s2. a1, ..., a7 are not retrieved, but contain additional evidence for the relevance or non-relevance of the respective subparts. Furthermore, in Fig. 4.4, a6 references d2. If we propagate the information coming from referenced objects as well, also d2 biases the RSV of s2.

We outlined how augmentation can support annotation-based structured document search. But of course not only structured documents can be the desired objects to retrieve, but also annotations, possibly as entry points into a discussion thread. If we only want to retrieve annotations, then in the example above, a3 could be augmented with a4 and a5 and the retrieval status value of a3(a4,a5) would be calculated. Note that a3 could also possibly augmented with content from s2 contained in a merged annotation target of a3.

4.3.1.2 Controlling the Augmentation Behaviour

The augmentation behaviour can be controlled by special commands. The expressions

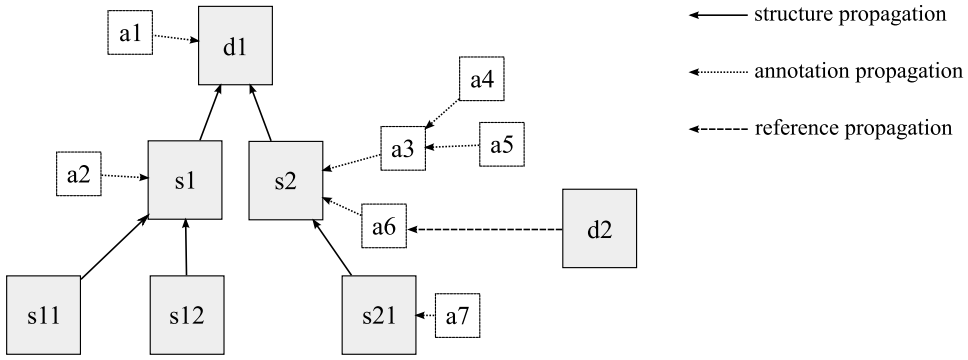


Figure 4.4: Annotation-based structured document retrieval. Grey boxes reflect subparts of a structured document, white boxes are annotations. Arrows denote structure, annotation and reference propagation, respectively.

```
_structure_propagation()
_annotation_propagation()
_reference_propagation()
```

say that we augment a context by its logical document structure, by annotations (including fragments and merged targets), and by referenced objects, respectively (which is the default behaviour). On the other hand,

```
_no_structure_propagation()
_no_annotation_propagation()
_no_reference_propagation()
```

omit the logical structure, annotations and referenced objects, respectively, from (knowledge and relevance) augmentation. By combining these special commands, we can fine tune the set of objects involved in augmentation. In the above example, if we say that no structure and reference propagation should be performed, the augmented context of s2 is s2(a3(a4,a5),a6). If we omit annotation and reference propagation, s2's augmented context is s2(s21). If we only disallow reference propagation, s2(s21(a7),a3(a4,a5),a6) is the augmented context of s2. If we allow all kinds of propagation, we gain s2(s21(a7),a3(a4,a5),a6(d2)) as the augmented context of s2.

4.3.1.3 Example

We will now discuss an example for document and discussion search in POLAR. Consider the following simple knowledge base consisting of annotations and structured documents:

```
document(d1)
subpart(s1)
annotation(a1)
annotation(a2)
d1[ 0.7 ir
    0.9 s1[ 0.6 db   0.5 *a1 ] ]
a1[ 0.4 t1< 0.6 db >
    0.75 is   0.6 *a2 ]
a2[ 0.4 t2< 0.75 is >
```

```
      0.9 ir ]
0.5 °ir   1.0 °is   1.0 °db
```

Document d1 is about information retrieval ('ir') and has a section (subpart) s1 about databases ('db'). s1 is annotated by annotation a1 which is about information systems ('is'). a1 is annotated by a2, which is about information retrieval again. a1 and a2 form a toy discussion thread. The merged target of a1 is determined by the content of its annotated object s1 (analogously for a2). 'ir' has an *idf* of 0.5, 'is' and 'db' both have an *idf* of 1. This knowledge base is the basis for our further considerations on structured document IR and discussion search.

Structured Document IR We want to perform structured document retrieval. This means our ranking should contain subparts and documents, but no annotations. Let us assume we search for documents about information systems.

```
q1[ is ]
relevant1(D) :- //D->q1 & document(D)
relevant1(D) :- //D->q1 & subpart(D)
```

The first line defines our query, the second and third line says that every document and subpart is relevant if its augmented context implies the query. Without knowledge augmentation, d1 and also s1 would not be retrieved, since they do not know about 'is'. But now, we gain:

```
?- relevant1(D)
0.375  (s1)     # from s1(a1)
0.3375 (d1)     # from d1(s1(a1))
```

The term 'is' is propagated from a1 to s1 (and has a weight of $0.5 \cdot 0.75 = 0.375$ in s1(a1). It is further propagated to d1 with a weight of $0.5 \cdot 0.75 \cdot 0.9 = 0.3375$ in d1(s1(a1)). These values are multiplied with the *idf* of 'is' to get the final result (yielding the same values again due to multiplication with 1). a1 and a2 are not retrieved, although their augmented contexts (a1(a2) and a2(t2), respectively) know about 'is', because they are not classified as being a subpart or document.

Discussion Search Based on our toy knowledge base we can also perform discussion search. Here, annotations are the focus of retrieval, and we exploit the discussion context of each annotation. Consider a new query searching for 'ir'; relevant are only annotations satisfying this query. We gain:

```
q2[ ir ]
relevant2(A) :- //A->q2 & annotation(A)
?- relevant2(A)
0.45 (a2)     # from a2(t2)
0.27 (a1)     # from a1(t1,a2)
```

'ir' has a weight of 0.9 in a2 and $0.6 \cdot 0.9 = 0.54$ in a1(t1,a2)[10]. These values need to be multiplied with the *idf* of 'ir', which is 0.5. Consider a third query, this time for databases:

[10] You may notice that a1's augmented context is not a1(t1,a2(t2)), as we would expect it from the discussion so far. t2 is the merged annotation target of a2 and contains content from a1, which should not be considered again. This peculiarity of annotations w.r.t. augmentation will be subject to discussion later.

```
q3[ db ]
relevant3(A) :- //A->q3 & annotation(A)
?- relevant3(A)
0.24 (a1)    # from a1(t1,a2)
```

a1 is returned because is annotates an object, s1, which is about databases. We find this information in a1's merged target t1, so we gain an overall RSV of $0.6 \cdot 0.4 = 0.24$ for a1. This example shows that discussion search can be influenced by items which are not part of discussion threads.

When performing discussion search like above, POLAR tries to find a suitable entry point in the discussion thread. This entry point is supposed to mark the beginning of the discussion about the given topic in the discussion thread[11]. It is the *best* entry point if we assume that the reader starts at this point and navigates through the replies in order to follow the whole discussion about the given topic from the beginning to the end.

4.3.2 Enriching a Document Ranking with Annotations

Consider the following POLAR knowledge base:

```
document(d1)
document(d2)
annotation(a1)
annotation(a2)
annotation(a3)

d1[ 0.7 ir
    0.9 *a1 ]
a1[ 0.4 is   0.2 ir]

d2[ 0.3 is *a2 *a3 ]
a2[ 0.5 ir ]
a3[ 0.2 ir ]

0.5 °ir   1.0 °is   1.0 °db
```

Now we seek all documents about 'ir':

```
q1[ ir ]
?- D->q1 & document(D)
0.377 (d1)    # from d1(a1)
0.300 (d2)    # from d2(a2,a3)
```

A system could produce a ranking with d1 and d2 and return this to the user. However, especially in the case of d2 a user might wonder why this item was retrieved. The idea is to present the user a ranking which groups all evidence from annotations by their annotated documents. To do so, an additional query is performed:

[11] Note that, in case of topic changes, a discussion about a topic can start somewhere deep within the discussion thread.

```
?- D[*A] & //A->q
0.09 (d1,a1)
0.25 (d2,a2)
0.10 (d2,a3)
```

The query returns all document-annotation pairs for documents which directly refer to a relevant annotation, ranked by the annotations' RSV and their access probability. From this evidence, a system might return to the user an annotation-enriched ranking like this:

1.	0.377	d1	
		0.09	a1
2.	0.300	d2	
		0.25	a2
		0.10	a3

Such a ranking would hint the user directly to relevant annotations, which she could use to determine if the corresponding document is relevant for her information need or not.

4.3.3 Document Access through Fragments and Highlighted Parts

We introduced fragments as special kinds of objects besides annotations and documents. Fragments can either be the main goal for retrieval, or they can help to find relevant documents. For instance, the query

```
?- F|| digital & libraries ||
```

finds fragments (i.e. annotated parts) about digital libraries. Such a query can be useful if, e.g., the user should be hinted to relevant highlighted parts when a document is displayed. A user is then able to find her own relevant highlighted parts more easily.

Fragments can be used to find relevant documents. One way is their role in knowledge augmentation described above. Another way is to search for documents having relevant fragments. Consider the case that the user remembers that she highlighted a passage about digital libraries in a document, and now wants to find the document again. The rule and query

```
relevant(D) :- D[ ||F ] & F|| digital & libraries ||
?- relevant(D)
```

finds documents containing fragments about digital libraries. A similar query can be posed based on probabilistic inference:

```
q[ digital libraries ]
relevant(D) :- D[ ||F ] & F->q
?- relevant(D)
```

Documents are ranked based on relevant fragments and their access probability. Documents having relevant fragments are ranked higher than documents with mainly non-relevant ones.

We could also ask for fragments annotated by Mike in a document d1:

```
relevant(F) :- d1[ ||F ] & F||*A|| & A.author(mike)
?- relevant(F)
```

4.3.4 Users and Groups

In a repository consisting of public, shared and private annotations, a user is not allowed to see every annotation. Private annotations should be visible only by the users who created them, and shared annotations can only be accessed by members of the appropriate group. We can represent users and groups, as they are contained in the annotation model introduced in the last chapter, as objects and attributes on the global database level:

```
group(infosystems)
group(interactive_systems)
peter.memberOf(infosystems)
thomas.memberOf(infosystems)
harold.memberOf(interactive_systems)
```

There are two groups, “infosystems” and “interactive systems”. Peter and Thomas belong to the “infosystems” group, while Harold is member of “interactive systems”. We further specify the authorship, associated group and scope of annotations:

```
a1.author(peter)
a2.author(thomas)
a3.author(harold)
a1.group(infosystems)
shared(a1)
private(a2)
public(a3)
```

Peter is the author of a1, Thomas the one of a2 and Harold the author of a3. a1 is a shared annotation and belongs to the group “infosystems”. a2 is a private annotation, while a3 is a public one. Our termspace, the three annotations and the query q are:

```
0.5 °collaborative_systems
a1[ 0.8 collaborative_systems ]
a2[ 0.6 collaborative_systems ]
a3[ 0.5 collaborative_systems ]
 q[ collaborative_systems ]
```

Let us say that Thomas is the current user of our system:

```
current_user(thomas)
```

Then we can define rules for relevant objects which are visible to Thomas:

```
visible(A) :- private(A) & A.author(U) & current_user(U)
visible(A) :- shared(A) & A.group(G) & U.memberOf(G) &
              current_user(U)
visible(A) :- public(A)
```

The first rule says that the current user can see his own annotations. He can also see annotations which are shared among groups he is a member of (second rule). And of course he can see all public annotations, as the third rule states. We return a ranking of documents which satisfy the query and which are visible to the current user (in this case Thomas):

```
?- A->q & visible(A)
0.4  (a1)
0.3  (a2)
0.25 (a3)
```

a1 is retrieved since it is a shared annotation and Thomas is member of the group a1 belongs to. a2 is retrieved because it is a private annotation and Thomas is its author. a3 is retrieved because it is a public annotation. All annotations are visible to Thomas, so nothing is filtered. Now consider the current user is Peter and not Thomas:

```
current_user(peter)
```

We then get:

```
?- A->q & visible(A)
0.4  (a1)
0.25 (a3)
```

a2 is not fetched since it is a private annotation and Peter is not its author. In case Harold is the current user, only a3 would be retrieved.

4.3.5 Semantic Annotations and Ontologies

One of the advantages of logic-based frameworks is the possible integration of additional knowledge into the retrieval function. Such external knowledge can consist of an ontology where classes and objects are semantically related to each other. A simple example can be an ontology regarding generalisation/specialisation (or "IS-A") relations among cities: each Hessian city (a city located in the German federal state of Hesse) is a German city, and each German city is a European city (similar relations can be identified for other European countries and their cities). Our toy ontology further says that each city located in Illinois is an American city. We can incorporate this ontology into our knowledge base as follows:

```
O[ german_city(C) ] :- O[ hessian_city(C) ]
O[ european_city(C) ] :- O[ german_city(C) ]
O[ american_city(C) ] :- O[ illinoisan_city(C) ]
```

This says that each object that is about a Hessian city is also about a German city and about a European city. The same holds for objects about cities in Illinois, which are also objects about American cities. Now consider that we have a document d1 about the city Darmstadt. Furthermore, consider a categoriser which infers with 0.7 probability that the "Darmstadt" mentioned in d1 is the Hessian city Darmstadt and with 0.3 probability that the American city Darmstadt, Illinois, is meant, and put this knowledge into a semantic annotation a1. The output of a document indexing and named entity recognition process could be

```
d1[ 0.8 darmstadt *a1 ]
a1[ 0.7 hessian_city(darmstadt)   0.3 illinoisan_city(darmstadt) ]
```

To search for all documents about the European city Darmstadt, we apply knowledge augmentation and pose the query

```
?- //D[darmstadt & european_city(darmstadt)]
```

The fact that Darmstadt is a Hessian city is known in the augmented context d1(a1) with 0.7 probability. Due to our ontology rules above, this is also the probability that our Darmstadt in d1 is a European city. The query returns d1 with a corresponding probability of $0.8 \cdot 0.7 = 0.56$.

4.3.6 Social Networks

With the advent of the so-called "Web 2.0", social community platforms (like Flickr, LibraryThing, Last.fm and YouTube) emerged which let users submit documents (textual documents, but also images and video). These documents can be shared among users, and, in turn, users can annotate these documents. Annotations can be textual comments, but also so-called *tags*. Collaborative tagging can be regarded as a kind of manual indexing of the document. Another key feature of social community platforms is the ability to maintain a list of *friends*. Friends lists are populated with friends a user knows from the real world, but can also be enriched with users sharing similar interests. For example, the social music platform Last.fm[12] calculates a similarity score between users, which is based on the musical taste. Another way of calculating the so-called *friendship similarity* is reported in Schenkel et al. (2008). By applying normalisation, such a score can be interpreted as a probability and thus be integrated into the POLAR framework.

Let us consider a toy knowledge base with four users, Frank, Eva, Paul and Martin. The friendship score is used to estimate the probability of the 'friend' attribute; for Frank, this might be:

```
0.8 frank.friend(eva)
0.1 frank.friend(paul)
0.1 frank.friend(martin)
```

Eva and Frank are close friends, while Paul and Martin are more strangers to Frank. Consider two documents, d1 and d2, which are tagged with 'pop' by annotations a1 (by Eva), a2 (by Paul) and a3 (by Martin):

```
d1[ 0.5 @a1 ]
d2[ 0.5 @a2   0.5 @a3 ]
a1[ pop ]
a2[ pop ]
a3[ pop ]

a1.author(eva)
a2.author(paul)
a3.author(martin)
```

(we see tags as meta annotations and access them with a probability of 0.5). Frank now wants all documents which are tagged with 'pop', and he prefers tags coming from his friends. In POLAR, this can be expressed with the rule and query

```
rel_soc(D) :- D[@A] & A[pop] & A.author(U) & frank.friend(U).
?- rel_soc(D)
```

For Frank, the tags provided by Eva are more valuable than the ones provided by Paul or Martin. So although d2 is tagged with 'pop' twice, and d1 only once, d1 is ranked ahead of d2:

```
0.4    d1
0.0975 d2
```

($0.4 = 0.5 \cdot 0.8$ and $0.0975 = 0.05 + 0.05 - 0.05 \cdot 0.05$). The probability of `rel_soc(d1)` (resp. `rel_soc(d2)`) is the *social score* of d1 (resp. d2) with respect to the given user (Frank).

[12] `http://www.last.fm/`

4.3.7 Ratings

Another interesting POLAR application are *ratings*. These often occur in commercial systems where users can rate, for instance, products or books. For example, within Amazon, users can rate (among other items) books and CDs by giving 0 to 5 stars (usually 5 stars means "very good" and 0 stars means "very bad") on a Likert scale. Such ratings are typical examples of meta annotations. We assume a 5-tier scale and map a rating onto the probability that the rated document is good. For this, we use the proposition 'rated_good'; its probability is determined by the rating: 0 means a probability of 0, 1 means a probability of 0.2, 2 means 0.4, 3 means 0.6, 4 means 0.8 and a rating of 5 means a probability of 1. (Note that this is a simple mapping of the scale to probabilities in order to show how such ratings can be applied in POLAR; actual applications might require a more sophisticated mapping.) Consider the following knowledge base:

```
d1[ 0.7 databases    0.5 @a1    0.5 @a2 ]
d2[ 0.8 databases    0.5 @a3    0.5 @a4 ]
a1[ 0.8 rated_good ]
a2[ 0.8 rated_good ]
a3[ 0.4 rated_good ]
a4[ 0.2 rated_good ]
```

We now seek for books about databases which are rated good:

```
rated_good(D) :- D[@A] & A[rated_good]
?- D[databases] & rated_good(D)
```

$P(\texttt{rated\_good(d1)}) = 0.5 \cdot 0.8 + 0.5 \cdot 0.8 - 0.5 \cdot 0.8 \cdot 0.5 \cdot 0.5 = 0.7$ and $P(\texttt{rated\_good(d2)}) = 0.5 \cdot 0.4 + 0.5 \cdot 0.2 - 0.5 \cdot 0.4 \cdot 0.5 \cdot 0.2 = 0.28$. The resulting ranking is

```
0.49  (d1)
0.224 (d2)
```

($0.49 = 0.7 \cdot 0.7$ and $0.224 = 0.8 \cdot 0.28$). d1 is ranked ahead of d2 due to the fact that it was rated better than d2.

4.3.8 Annotation-based Trustworthiness

We previously discussed the effect of negative content annotations on the probability that a term is *true* in an augmented context. We interpreted this scenario so that by knowledge augmentation, we decrease the trust in a1 being a good source for statements about 'political reasons'.

Especially in public discussions the question arises whether we can trust an annotation. Authors of annotations can simply be wrong or just talk nonsense. One measure of the *trustworthiness* is the number of positive or negative replies a comment gets. If there are mostly negative replies, we should not trust a comment; if there are mostly positive ones, then the comment is a trustworthy source for new information. Consider the following knowledge base:

```
0.7 °football
a1[ 0.7 football  0.5 -⋆a3   0.5 -@a4 ]
a2[ 0.5 football  0.5 ⋆a5    0.5 ⋆a6 ]
a3[]   a4[]   a5[]   a6[]
```

a1 talks about 'football' with high probability, but earned many negative comments (one on the content and one on the meta level). The probability that a2 is about 'football' is less, but it received two positive replies. Which comment is more relevant to a user, if we not only consider topical relevance, but also trustworthiness? We first say that an object is relevant if it is topically relevant to a query for 'football':

```
q[ football ]
topically_relevant(O) :- O->q
?- topically_relevant(O)
0.49 (a1)
0.35 (a2)
```

($0.7 \cdot 0.7 = 0.49$ and $0.7 \cdot 0.5 = 0.35$). The topical relevance would rank a1 ahead of a2. We add the following facts and rules to our program:

```
0.6 unconditional_trust(a1)
0.6 unconditional_trust(a2)
trustworthy(O) :- unconditional_trust(O)
trustworthy(O) :- O[*A]
trustworthy(O) :- O[@A]
!trustworthy(O) :- O[-*A]
!trustworthy(O) :- O[-@A]
```

The first two lines say that we trust a1 and a2 unconditionally with 0.6 probability. This value is important, because otherwise annotations which are not annotated again are not trustworthy. So line 3 says that an object is trustworthy if we unconditionally trust it. The next two rules say that an object is trustworthy if it has positive annotations. The last two lines say that it is not trustworthy if it has negative annotations. We get

```
?- trustworthy(O)
0.9  (a2)
0.15 (a1)
```

0.9 is the probability coming from the unconditional trust in a2 (0.6) and the two positive annotations a5 (0.5) and a6 (0.5). With the inclusion-exclusion formula, the probability that a2 is trustworthy is

$$0.5 + 0.5 + 0.6 - (0.5 \cdot 0.5 + 0.5 \cdot 0.6 + 0.5 \cdot 0.6) + 0.5 \cdot 0.5 \cdot 0.6 = 0.9$$

The probability that we have positive evidence for a1's trustworthiness is only coming from the unconditional trust in a1 (0.6). The probability for negative evidence about a1's trustworthiness comes from the annotations a3 (0.5) and a4 (0.5) and is $0.5 + 0.5 - 0.5 \cdot 0.5 = 0.75$. a1 is trustworthy if we have positive and no negative evidence about its trustworthiness; the probability of this event is $0.6 \cdot (1 - 0.75) = 0.15$. We see that we can trust a2 more than a1, based on the annotations. We now say that an object is relevant if it is topically relevant and trustworthy:

```
relevant(O) :- topically_relevant(O) & trustworthy(O)
?- relevant(O)
0.315  (a2)
0.0735 (a1)
```

with $0.315 = 0.35 \cdot 0.9$ and $0.0735 = 0.49 \cdot 0.15$. We gain a different ranking if we include annotation-based trustworthiness into our retrieval function.

4.3.9 Access Probability

Due to the fact that documents and annotations are not regarded as atomic objects in POLAR, but their context determined by the document structure, annotations and referenced objects is considered, the *access probability* plays a central role. When performing augmentation, the access probability can be compared to a propagation factor which controls to which degree terms, classifications, attributes, RSVs and their corresponding weights are propagated from subcontexts to (augmented) supercontexts. For structure queries, the access probability is used to provide a ranking of matching objects. Access probabilities have also been exploited for the determination of the trustworthiness of annotations.

The estimation of access probabilities is subject to the actual application on the one hand and, when used as a propagation factor for augmentation, subject to experiences made in experiments on the other hand. There are two basic views on the determination of access probabilities:

- In the *user-centric view*, access probabilities may be influenced based on user (i.e. reader) preferences. For example, a user may not want to consider annotations made by certain authors, or she gives a certain author more priority and therefore raises the access probability to annotations by this author.
- In the *system-oriented view*, the access probability is not determined by a user, but is based on statistics or a certain underlying model. For instance, the random surfer model (Page et al., 1998) assumes that Web links are randomly accessed with the same probability, which is determined by the number of links. To adopt a similar behaviour to POLAR and annotations, we may calculate the probability that a accesses its successor a' as
$$P(acc(a, a')) = \frac{1}{\#\text{annotations}}$$
where #annotations is the number of annotations of a. A further estimation of access probabilities is derived from experiments where we try to determine for which global access probability a retrieval functions yields the best results.

Both views might be mixed; the system might perform an initial estimation of the access probabilities, and then the user biases this value based on her preferences.

4.4 Related Work

By providing means to calculate $P(d \rightarrow q)$, POLAR follows the notion of retrieval as probabilistic inference proposed by van Rijsbergen (1986). As shown in Fig. 4.5, POLAR combines concepts and methods from areas such as hypertext and structured document (XML) retrieval (including modelling and querying of complex objects) and discussion search. Naturally, POLAR is related to other work from annotation-based IR.

4.4.1 Hypertext and Structured Document IR and Discussion Search

4.4.1.1 Hypertext IR

Graph-based Approaches Lucarella and Zanzi (1996) propose a graph-based object model for hypermedia documents. This model deals with objects and classes, attributes, and properties. Schema graphs can be defined on the object level; instance graphs are based on schema

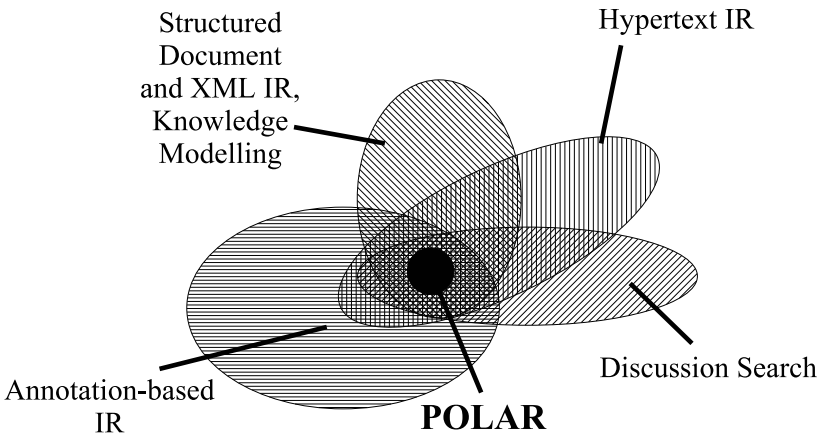


Figure 4.5: POLAR and related work

graphs on the object level. Retrieval is supported by means of so-called perspectives, which are subgraphs of the schema and instance graph; certain operations are offered to let the user specify conditions which objects of a select class have to meet. Other operations allow for object access and the combination of perspectives. Another interesting approach is reported by Chiaramella and Kheirbek (1996) who propose an integrated model for hypermedia and information retrieval based on conceptual graphs. Content knowledge contains concept types (the domain knowledge including generalisation/specialisation relations among concepts) for indexing documents, and structural knowledge contains the logical document structure of objects as well as possible relations among them. Complex queries can be posed to the underlying graph structure, and $P(d \rightarrow q)$ is calculated during query evaluation in order to create a ranking of documents. Both graph-based approaches are interesting for structured annotation hypertexts and annotation-based retrieval, since very complex structure and content queries are supported. Both approaches are different than the one presented here; for example augmentation, as we know it from POLAR, is not supported.

Propagation-based Approaches Another type of hypertext retrieval approaches takes neighbouring nodes (i.e., documents) into account when calculating a final RSV for a given node. The following approaches have in common that first a RSV for each node is calculated (applying, for example, a retrieval function based on $tf \times idf$), which, similar to relevance augmentation, is then propagated to the node whose final weight has to be determined. Frisse (1988) adds the normalised final weight of direct neighbours (which again consume the weight of their direct neighbours) to create a node's final weight. Frei and Stieger (1994) refine Frisse's approach by introducing the concept of *spreading activation*. *Constrained spreading activation* is based on a sophisticated link description which takes the link type, the content of the destination node and it neighbouring nodes, and link annotations into account. The similarity of a link description to the query is used in a decision phase to decide whether a link (and probably its subsequent ones) should be followed or not. If a link is followed, the normalised RSV of the destination node is added to the RSV of the original node using a propagation factor. In contrast to that, *weighted spreading activation* does not know a decision phase (all links are followed), but uses the similarity of the link to the query as an additional factor to be added to the final RSV. Both variants of spreading activation can utilise relevance feedback to improve

link descriptions. Experiments show that both spreading activation approaches outperform retrieval methods not considering any neighbouring nodes. Spreading activation motivated the relevance augmentation approach in POLAR.

Inference-based Approaches Certain other retrieval models include the structural context (logical structure, hypertext links or thread structure) in different ways to produce a ranking. For example, Croft and Turtle (1989) propose a retrieval model based on *Bayesian inference networks* for hypertext retrieval. Nodes represent documents, concept and the query, as in the standard Bayesian inference model reported in Turtle and Croft (1990). In order to deal with various links in hypertexts as well as thesaurus relations, dependencies between nodes (representing hyperlinks) and concepts (representing thesaurus links) are introduced to the inference network. A deductive database approach based on probabilistic logics and possible world semantics is probabilistic Datalog (Fuhr, 2000). Hypertext links can be considered in form of rules, whereas probabilistic facts represent weighted index terms.

Web IR Beitzel et al. (2003) describe an approach which makes use of a rich representation of Web documents, which includes, among others, the full text, titles, headers and anchors text of referring pages. The evidence coming from these components is merged to calculate a combined RSV. The approach is similar to the idea of augmentation applied here. Especially anchor texts are interesting, since they are similar to annotations due to the fact that they directly refer to the page a link points to. Following the ideas of Page et al. (1998) and Kleinberg (1998), where the link structure is mined to extract evidence if a page is a good authority, also POLAR basically supports the exploitation of such non-topical evidence found in the link structure.

Further hypertext approaches and a thorough discussion on hypertext IR can be found in Agosti and Smeaton (1996).

4.4.1.2 Structured Document Retrieval

The hypertexts which are the underlying data structure of the approaches mentioned above are not necessarily reflecting the logical structure of documents. Above approaches can thus be applied on a wide range of hypertext containing structural links as well as inter-document links (like, e.g. bibliographic references). In the 90s and especially with the emergence of XML as a language to represent structured documents as well as data items, more methods focused on *structured document retrieval*.

Structured documents are hypertexts as well, but with certain peculiarities. The report by Chiaramella et al. (1996) presents a model for *hypermedia documents* (sometimes also referred to as *complex objects*) most approaches for structured document and XML retrieval are based upon. One important aspect of structured documents is that document components or subparts are nodes in a hypertext connected by structural links reflecting the logical document structure. This way, such links establish an *aggregation* relation. Consider, for example, this thesis which consists of parts, chapters, sections, subsections, subsubsections and finally paragraphs, code snippets, figures and tables. Chapters aggregate sections and paragraphs, parts aggregate chapters, and the whole thesis is an aggregation of chapters and parts. So in the hypermedia model, the thesis as well as each part, each chapter etc. would be nodes in a hypertext which are connected by structural links. Besides structural links, the model by Chiaramella et al. also allows for other (intra- and inter-document) links; examples are references to other document parts or the bibliographic references pointing to external documents. The document

model in POLAR, where we define documents as contexts, basically follows and extends this hypermedia model. Both documents and annotations are regarded as structured documents whose aggregation is modelled by the subpart relation. Knowledge augmentation aggregates such subcontexts into supercontexts. POLAR goes beyond structured document retrieval by also incorporating annotations into the aggregation.

As outlined before, POLAR is strongly related to POOL (Rölleke and Fuhr, 1996; Fuhr and Rölleke, 1998; Rölleke, 1998; Fuhr et al., 1998). POOL is targeted at modelling complex hypermedia objects and providing means for structured document retrieval. POLAR integrates all POOL concepts like four-valued logics, probabilistic logics, object-oriented modelling with terms, classifications and attributes, as well as aggregation of objects by means of access to subcontexts allowing for knowledge augmentation.

Another knowledge modelling approach, also targeted at hypermedia and structured document retrieval, is MIRTL (Meghini et al., 1993). MIRTL is based on terminological logic. It is good at representing complex objects and complex queries, but lacks direct support of important IR features like term weights and the calculation of a retrieval status value.

In recent years, many approaches for structured document retrieval were developed which utilise the XML representation of the document base. The logical structure of documents is given by the document type definition (DTD) or an XML schema definition; XML documents basically adhere to the hypermedia model above. XML retrieval emerged as a branch of structured document retrieval. Many standard retrieval approaches were adapted in order to consider the structural context of documents. For instance, Piwowarski et al. (2003) utilise Bayesian inference networks for structured document retrieval to find a best entry point within the document structure. They apply a flat retrieval function which calculates an initial RSV for each document node and then use Bayesian inference to determine a best entry point. Other approaches use *language models* for structured document retrieval (see, e.g., Ogilvie and Callan (2004)). Abolhassani and Fuhr (2004) extend the *divergence from randomness* approach by incorporating a so-called third normalisation which introduces the level of a node in a structured document (with a level of 1 for root node) as an additional parameter to the Inf_2 function. This makes the risk of accepting a term higher for lower levels, penalising nodes which are not specific. Fuhr and Großjohann (2004) present XIRQL, an XML query language which addresses the problem that common XML query languages do not offer any support for IR-oriented XML querying. Within XIRQL, it is possible to pose content-and-structure queries to XML documents. Specificity-oriented search, aiming at returning only the most specific document nodes, is supported by an augmentation strategy very similar to POLAR's knowledge augmentation.

More approaches for structured document (especially XML) retrieval are reported in the proceedings of INEX, the Initiative for the Evaluation of XML Retrieval (Fuhr et al., 2003, 2004, 2005, 2006, 2007). These proceedings also contain a thorough discussion about the task itself and suitable evaluation measures.

4.4.1.3 Discussion Search

Discussion threads are made of newsgroup articles, emails or, as we have shown before, nested annotations. In Section 4.1.4.1 we therefore identified *discussion search* as an important task within annotation-based IR. The goal of discussion search is to find documents (newsgroup articles, emails or annotations) whose own content satisfies the query. Focusing on a document's "own content" means that parts belonging to previous messages (in form of quoted text in emails of newsgroup articles) are not considered as part of the document. Because POLAR also aims at

supporting annotation-based discussion search, it is worth looking at related discussion search approaches.

Xi et al. (2004) propose a feature-based discussion search approach based on linear regression and support vector machines, respectively. A message's content features (9 in total) comprise the content of the message itself, the title's content, the content of the root of the thread, the ancestor, children, etc. For all these features, a ranking score is calculated according to 3 different ranking functions, so a document description contains 27 ranking scores (one score for each combination of content features and ranking function). Some further features like the number of descendants are added to the document description, together with certain author-dependent features. Support vector machines and linear regression is used to train a function which returns the final RSV for a message. While feature-based methods usually performed well in IR, and the present one is no exception, their disadvantage is that they need a training sample (documents with training queries and corresponding relevance judgements) to learn the desired function.

Another approach not relying on a training sample utilises language models and thread-based query expansion for discussion search (Balog et al., 2006). Discussion search was a major task in the Enterprise Track of the Text Retrieval Conference (TREC) in 2005 and 2006. The corresponding TREC proceedings (Voorhees and Buckland, 2005, 2006) introduce further discussion search methods.

4.4.2 Annotation-based IR

While the approaches explained before can potentially be applied for annotation-based document and discussion search, none of them addresses annotations directly. In fact, surprisingly few annotation-based retrieval methods came up during the last decade.

Fraenkel and Klein (1999) identify annotations as an important source for retrieving relevant texts, as they show with examples coming from the Bible and the Talmud. Their focus is on the question how annotations are properly embedded in the main text in order to allow for proximity search or, in other words: "How should we measure the distance between a word in an embedded annotation and a word that occurs later in the main text?". They discuss three possible alternatives and corresponding implementation issues.

Golovchinsky et al. (1999) derive and expand full-text queries with terms contained in annotated passages (the fragments in POLAR), based on the assumptions that these passages reflect users' interests more accurately. They compared the annotation-generated queries (considering annotated fragments only) with those expanded by relevance feedback (considering whole relevant documents). Experiments show that annotation-generated queries perform better than those derived by relevance feedback. This result confirms the assumption that annotated fragments reflect users' interests quite well, and it pretty much motivated the introduction of fragments into the POLAR framework.

Agosti and Ferro (2005) describe an approach for annotation-based document retrieval using fusioning techniques. The underlying data structure is similar to the structured annotation hypertext presented in Chapter 3; in fact, the definition of a structured annotation hypertext was inspired by Agosti and Ferro's definition of a document-annotation hypertext. In their approach, a compound similarity score for annotations with respect to the given query is calculated first, which recursively combines the score of an annotation itself and the score of its successor nodes in the annotation thread. This score is used to calculate a document score based on annotations alone. The annotation service knows all about annotations, their content and metadata, and to which document they belong to, but nothing about the document content

itself. To get the content-based score of a document, the query is passed to the corresponding digital library management system, which returns a document score using its own retrieval routines. The document's annotation-based score is then combined with its content-based score by applying certain fusioning techniques. The data fusioning approach is supposed to be integrated in the Flexible Annotation Service Tool (FAST), which offers an API for annotation-based document retrieval as well as basic database- and content-oriented annotation querying functions (Agosti and Ferro, 2006). Once integrated into FAST, the data fusioning approach can handle the realistic situation that annotation services and document management services are distributed and annotations are stored independently from the document repository. However, the approach lacks means to deal with some peculiarities of annotations, like positive and negative annotations or meta and content ones. As an outlook, POLAR might be a candidate to implement some of the database- and content-oriented annotation querying methods described in (Agosti and Ferro, 2006).

Cabanac et al. (2007) present an interesting annotation approach which focuses on the *social validation* of annotations. The underlying assumption is that an annotation makes sense when a social group judges so. Such judgements are more or less implicitly contained within annotation-based discussion threads. An annotation is socially validated when its arguments are confirmed or refuted. The social validity measures the degree of confirmation or refutation. This value is recursively calculated with the replies to an annotation and their social validity. It utilises an in-depth analysis of objective and subjective data about annotations, deriving important measures like the annotator's expertise, whether she provided references and how many, the comment type (modification, question or example) and the semantics of the annotator's opinion (confirmation and refutation). Cabanac et al. discuss three different methods to calculate social validity. Social validity can support annotation-based retrieval in different ways; for example, the content of a positive and socially validated annotation qualifies the annotated objects as being trustworthy, whereas negative annotations attack the importance of the annotated objects. POLAR basically supports these mechanisms: in augmentation, negative annotations decrease term weights or retrieval status values, respectively, of the annotated objects, while positive annotations increase them. We further considered positive (confirming) and negative (refuting) annotations when talking about the trustworthiness of annotations in Section 4.3.8. Nevertheless, the model proposed by Cabanac et al. provides a more sophisticated view on social validity which can potentially be covered by modelling appropriate rules and facts in POLAR.

4.5 Summary and Discussion

This chapter presented annotation-based knowledge modelling, querying and retrieval with the POLAR framework. First an overview of information retrieval in general was given, and then the POLAR framework was presented. POLAR is an extension of POOL; while the latter focuses on modelling complex objects and on structured document retrieval, the former extends this by introducing new concepts known from structured annotation hypertexts. In POLAR, documents and annotations are complex objects which establish so-called contexts. Propositions (terms, attributes and classifications) can be made within such contexts. For example, the occurrence of a term in a document is a proposition made in the document's context. POLAR copes with different subcontext types, established by subparts, annotations, fragments, merged targets and references. Within the global database contexts, assertions can be made about objects (e.g., metadata) and global classes and IS-A relations can be modelled.

POLAR offers certain expressions to query the underlying structured annotation hypertext. It further supports the calculation of the implication probability $P(d \rightarrow q)$. POLAR can deal with four truth values, which is important for example to reflect contradictions in discussions. A further important annotation attribute supported by POLAR is the polarity. One of POLAR's core concepts is knowledge augmentation, where a context is extended with the knowledge in its subcontexts. For instance, the context of a document can be augmented with its annotations. Knowledge augmentation combined with calculating $P(d \rightarrow q)$ allows for annotation-based structured document retrieval on the one hand and is a tool for discussion search on the other. Similar to knowledge augmentation, relevance augmentation propagates retrieval status values from subcontexts to supercontexts. This strategy is more suitable if there is no access to the document knowledge in distributed sources, but only to retrieval status values w.r.t. the given query. Application examples show the flexibility and expressiveness of POLAR, its powerful querying and retrieval facilities and the ability to exploit even non-topical information contained in structured annotation hypertexts. While POLAR is aimed at annotation-based retrieval, it combines and extends concepts known from hypertexts and structured document retrieval as well as discussion search.

With POLAR, a framework is created which supports various information needs arising when dealing with annotations. The goal was not only to define yet another solution to retrieve documents with the help of annotations (although this is the main focus of POLAR), but to provide advanced knowledge modelling and querying mechanisms. A logic-based solution was chosen since it allows for easy integration of additional knowledge into the retrieval process, as well as modelling and querying the rich representation established by annotations and documents. It further allows for the creation of flexible retrieval functions which are able to fully exploit the information contained in structured annotation hypertexts which would otherwise be ignored by "thinner" retrieval approaches. We cannot just reuse POOL for the task at hand since by nature, POOL can only deal with trees as we find them when dealing with the logical document structure. Modelling structured annotation hypertexts in POOL is not impossible, but would be a tedious task. Only subsets of structured annotation hypertexts which have a tree-like structure (e.g., documents with connected annotation threads, without references and only allowing one annotation target) are supported by POOL directly. A document and a connected annotation thread would then be regarded as a complex object in POOL. But even then certain other crucial concepts like negative annotations or the distinction between content and meta level annotations could only be supported indirectly with POOL (if at all).

The analogy of knowledge augmentation is that of a reader who reads the content of an annotation or document and likes to know more. If it is not clear what the annotation is about, the reader would refer the annotated parts to grasp the context an annotation was made in. Knowledge augmentation does the same by accessing merged targets. The reader would access content annotations to get different opinions about the topics or additional information or interpretations – this is analogous to knowledge augmentation accessing annotations. The reader would probably also consider highlighted parts as more important than non-highlighted ones, so these fragments would raise her attention in a natural way. POLAR tries to reflect this by accessing fragments during knowledge augmentation. If another object is referenced, a reader would access it as well with a certain probability to gather additional information. This is similar to knowledge augmentation following links to referenced objects. The exception are meta annotations, which are excluded from knowledge augmentation since this would mix up different levels – the meta and the content level.

POLAR contexts contain information about which documents are referenced or by which objects the document or annotation is annotated. Fragments give information about important passages and merged annotation targets contain the parts of the objects an annotation refers to. For example,

```
a1[ t1<...>   f1||...*a2||   @a3
    a11[...]  =>d1    ...]
```

models an annotation a1 (content not given here) as a complex object containing a subpart a11. It has a merged annotation target t1, a fragment f1 (annotated by a2) and a meta annotation a3. a1 references d1. Knowledge and relevance augmentation would now augment a1 to create the augmented context a1(t1,f1,a2,a11,d1). Propositions from annotations, annotated parts, fragments, subparts and referenced objects are propagated to the augmented context of a1, so the current design of POLAR determines the direction in which knowledge is propagated when performing augmentation. Since POLAR is an open system, possible extensions might be introduced in the future. For example, instead of merging annotation targets and seeing them as one virtual object, we might introduce each annotation target as an own subcontext. Or, in addition to modelling which other objects an object references (a1 references d1 in the example above), we might model directly if an object is referenced by another one (in the context d1, we would then directly model that it is referenced by a1, which is not possible right now). For the time being, POLAR supports propagation in the given directions, since the described extensions would also mean more complex semantics and implementation.

In this chapter, we introduced POLAR informally and gave some application showcases. The next chapter brings a formal definition of POLAR's syntax and semantics.

5

POLAR Syntax and Semantics

Alles Gescheite ist schon gedacht worden, man muß nur versuchen, es noch einmal zu denken.

(Johann Wolfgang Goethe)

In the last chapter, we informally introduced the POLAR framework. We have seen that POLAR allows for complex queries to the underlying structured annotation hypertext, implementing advanced retrieval algorithms for annotation-based document and discussion search.

In this chapter, we are now going to specify POLAR's syntax and semantics. We start with a discussion of POLAR's syntax. Based on the syntax, we give a semantic interpretation of POLAR expressions. The semantics of POLAR is based on the notion of possible worlds and modal logics. After presenting POLAR's semantics, we discuss retrieval functions based on uncertain inference, precisely the probability $P(d \rightarrow q)$ that a document d implies a query q.

5.1 Syntax

We will now describe the syntax of POLAR programs. First, we start with the basic syntax, which mainly describes the constructs used for modelling extensional knowledge. We then continue with the syntax of rules and queries. To describe the POLAR syntax, we use the extended Backus-Naur-Form (EBNF), as specified in (International Organization for Standardization, 1996).

5.1.1 Basic Expressions

Figure 5.1 shows the basic syntax of POLAR programs, which we will now describe in more detail.

A *program* consists of one or more *clauses* which can be a fact, a query, a rule, a context or a predicate. Queries and rules are described in more detail in the next subsection.

A *fact* can be a proposition, the negation of a proposition (denoted by a preceding '!'), or a proposition with a weightlist. A *proposition* is a term, a classification (determined by the class name and the ID of the classified object) or an attribute (consisting of the object the attribute belongs to, the attribute name and the attribute value as a constant). A constant might be an object ID, a number or any arbitrary string (all three are not further specified here). A *weight list* consists of up to four weights ranging from 0 to 1. The first weight stands

```
program          = clause {program};
clause           = fact | query | rule | context | predicate;
context          = obj-id "["
                     {mergedtarget} contextprogram "]";
mergedtarget     = {weight} obj-id "<" factlist ">";
contextprogram   = contextclause {contextprogram};
contextclause    = fact | subpart | annoref | reference
                 | fragment;
subpart          = {weight} obj-id "[" contextprogram "]";
fragment         = {weight} obj-id "||" factlist {annoref} "||";
factlist         = fact {factlist};
fact             = proposition
                 | "!" proposition
                 | weightlist proposition;
annoref          = contentannoref | metaannoref;
contentannoref   = {weight} "*" obj-id
                 | {weight} "-*" obj-id;
metaannoref      = {weight} "@" obj-id
                 | {weight} "-@" obj-id;
reference        = {weight} "=>" obj-id;
proposition      = term
                 | classname "(" obj-id ")"
                 | obj-id.attr-name "(" constant ")";
predicate        = {weight} term-predicate
                 | {weight} class-predicate
                 | {weight} attr-predicate;
term-predicate   = "°" term;
class-predicate  = "°°" classname;
attr-predicate   = "°°°" attr-name;
constant         = obj-id | number | string;
weightlist       = weight
                 | weight "/" weight
                 | weight "/" weight "/" weight
                 | weight "/" weight "/" weight "/" weight;
```

Figure 5.1: Basic POLAR syntax. See Fig. 5.2 for rules and queries.

for the probability of a proposition to be *true*, the second weight for *false*, the third one for *inconsistent* and the last one for *unknown*.

Each POLAR object is modelled as a *context*. All such contexts consist of an object ID identifying the object or clause, respectively. Each context may contain a merged target. A *merged target* is a special context containing facts appearing in each target an annotation has. Like any other context, a merged target is uniquely identified by an object ID.

There is a specific context program for each context, which is different from general programs. A *context program* contains *context clauses*, which can be facts, subparts, references to annotations (so-called annorefs), references to other objects, or fragments. *Subparts* are the subcomponents of structured objects; they do not have any merged target.

A *fragment* is another special context which is identified by an object ID. Since fragments are areas created during the process of annotation, they are either linked to an annotation in case they are the target of this annotation, or they can be the object a reference points to.

In the former case they are strongly connected to exactly one annotation. In the latter case they are not an annotated object. Therefore, fragments can contain at most one reference to an annotation (annoref). An *annoref* denotes the reference to an annotation which annotates the given context, and can be either a contentannoref or a metaannoref. A *contentannoref* denotes the object annotating a context on the content level, whereas a *metaannoref* points to an object annotating a context on the meta level. Both contentannoref and metaannoref may have associated weights which determine the probability that the annotation is accessed from the annotated context. An annoref can indicate that an annotation is negative w.r.t. the given context (preceding '-' sign). A *reference* denotes the fact that a context refers to another object. Such references are used if annotations link two objects. Like annorefs, references also may have a weight as access probability.

Merged targets, fragments and subparts are *subcontexts* of the context they appear in. Annorefs and references turn the contexts they point to to subcontexts of the contexts they belong to.

Predicates contain global values about the importance of specific terms, classes or attributes. Although these values might be derived from collection statistics, in case of terms for example by calculating the inverse document frequency, they may also be given explicitly by the user. So in POLAR we offer means to define these weights explicitly. We distinguish between *term predicates*, *class predicates* and *attribute predicates*. Predicates are further discussed in Section 5.3.

5.1.2 Rules and Queries

Figure 5.2 shows the syntax of POLAR rules and queries. We will give a detailed description now. A *rule* consists of a head as a goal and a subgoal list. A *head* may be a context goal with an object and a so-called fact goal, or only a fact goal. An *object* in this syntax description can be an object ID or a variable in place of an object ID. By convention, variables start with capital letters, whereas object IDs start with small letters. A *fact goal* is an atom or its negation. *Atoms* can be terms, classnames with a corresponding object (object ID or variable), or attributes consisting of an object ID or variable, respectively, an attribute name and an attribute value, which can be either a variable, an object ID or a constant. Similar to POOL (Rölleke, 1998, p. 78), we avoid recursive context goals – the structured annotation hypertext cannot be described by rules.

A *query* is a subgoal list preceded by the query mark "`?-`".

A *subgoal list* is a list of subgoals outside contexts and separated by a '`&`'. There are many possible kinds of subgoals. The first variant is a context clause with an instruction to perform (knowledge) augmentation ('`//`'). It consists of an object (again, an object ID or a variable) and a special context subgoal list called *csubgoal list*. The second variant is composed of an object and a context subgoal list only. The third kind of subgoal describes fragments as subgoals, containing a *fsubgoal list* (special subgoal list for fragments). The next two variants describe a retrieval function based on uncertain inference, where we seek for (augmented or non-augmented) objects implying another object. Finally, a fact goal can be a subgoal as well.

Csubgoal lists are lists of csubgoals, which are another special kind of subgoal, here within contexts. We introduce csubgoals because different kinds of subgoals are possible within contexts but not outside them. A *csubgoal* can be an annoref goal, a reference goal, a fact goal or a fragment goal. *Annoref goals* describe objects annotating another object. We distinguish between content and meta level annotations. For instance, "`d[*A]`" stands for all content level annotations of d; "`D[*a]`" denotes all objects annotated by the content level annotation a.

```
rule          = goal ":-" subgoallist;
query         = "?-" subgoallist;
head          = object "[" factgoal "]" | factgoal;
subgoallist   = subgoal {"&" subgoallist};
subgoal       = "//"  object "[" csubgoallist "]"
              | object "[" csubgoallist "]"
              | object "||" fsubgoallist "||"
              | object "->" obj-id
              | "//"  object "->" obj-id
              | factgoal;
csubgoallist  = csubgoal {"&" csubgoallist};
csubgoal      = annorefgoal |  referencegoal
              | factgoal | fragmentgoal;
fsubgoallist  = fsubgoal {"&" fsubgoal};
fsubgoal      = factgoal | annorefgoal;
annorefgoal   = "*" object | "@" object |
                "-*" object | "-@" object;
referencegoal = "=>" object;
fragmentgoal  = "||" object;
factgoal      = atom | "!" atom;
atom          = term
              | classname "(" object ")"
              | object.attr-name "(" attr-value ")";
object        = obj-id | variable;
attr-value    = obj-id | variable | constant;
```

Figure 5.2: Syntax of POLAR rules and queries

Similarly, *reference goals* describe objects referencing or being referenced by another object. *Fragment goals* are used to look for fragments of an object or the object belonging to a given fragment. *Fsubgoal lists* are similar to csubgoal lists, but only contain an annoref goal and fact goals.

5.2 Semantics

In the last section we defined the syntax of POLAR.Now, we are going to discuss the semantics of POLAR programs. We define formal semantics for two reasons: first, we can check if two POLAR programs are equivalent by showing that each model of the first program is also a model of the second one, and vice versa. Second, we are able to show whether or not an implementation of POLAR behaves correctly.

The model is based on the notion of *possible worlds*, which we will explain first. Subsequently we define the semantics of basic POLAR programs. Afterwards we discuss knowledge augmentation, i.e. the semantics of augmented contexts. Finally, we continue with queries and rules.

5.2.1 Possible Worlds

5.2.1.1 Introduction

Our formalism to describe the semantics is based on the *possible worlds model* as discussed by Fagin et al. (1995). The basic idea is that of so-called *agents* which consider certain worlds possible. Such *possible worlds* contain a set of propositions, which can be, in case of classic two-valued logics, either *true* or *false* in a world. For example, consider two propositions φ_1 and φ_2, then we obtain four possible worlds: $w_1 = \{\varphi_1, \varphi_2\}, w_2 = \{\varphi_1\}, w_3 = \{\varphi_2\}$ and $w_4 = \{\}$. In world w_1, both propositions are true, whereas in w_2, only φ_1 is *true* and the closed world assumption applies that φ_2 is *false* (since it is not *true*). In w_3, φ_1 is *false* and φ_2 *true*. In w_3, both propositions are *false*. It is now said that an agent K *believes* a proposition φ to be true if φ is *true* in all worlds it considers possible from the current world. If, for example, the agent K is in the (not further specified) world w_0 and considers w_1 and w_2 possible, it knows that φ_1 is true, but has somewhat inconsistent information about φ_2. Instead of saying "an agent considers a world w possible", we also say "an agent can *reach* or *access* a world w". The worlds an agent K considers possible from the current world are determined by *accessibility relations* which state which worlds can be reached from a starting world by a certain agent.

Consider a real life example from the German football league Bundesliga, where Dortmund and Schalke are famous teams. Let φ_1="Schalke won" and φ_2="Dortmund won" and $w_1, \ldots, w_n$ as above. If Schalke and Dortmund did not play against each other and the matches are over, all four worlds $w_1, \ldots, w_4$ are possible and furthermore, our agent K is currently in one of these four worlds (but may not know in which one in case the agent has not heard of the results yet). So the agent considers all four worlds possible, which means that he actually does not *know* if Schalke or Dortmund have won – in the case of Schalke, for example, our agent considers worlds possible where Schalke won or not. If the agent would only consider worlds w_1 and w_2 possible, where "Schalke won" is true, maybe because somebody told him the result of the Schalke match, he would *know* that Schalke won, since this proposition is *true* in each world the agent considers possible (can reach). But he would not know whether Dortmund won, since the cases that Dortmund won and did not win are possible. If Dortmund and Schalke played against each other, the case that both won is impossible; either one of them won or they played a draw. So without further knowledge, our agent it either in w_2, w_3 or w_4, but not w_1, and considers w_2, w_3 or w_4 possible[1].

The possible worlds model above can be extended by introducing probabilities as proposed by Nilsson (1986). Here, each world has a probability assigned and the probability that a proposition φ is true is determined by the sum of probabilities of all worlds in which φ is true (given that all worlds are mutually exclusive). If we take our example above, let us assume that from a (not further specified) world w_0, our agent K considers each world possible with a certain probability, for instance $P(w_1) = 0.3, P(w_2) = 0.4, P(w_3) = 0.2$ and $P(w_4) = 0.1$ (see Figure 5.4). If we refer to our Bundesliga example again (with Schalke and Dortmund not playing against each other), then one of the four worlds must be actually occurring, and all worlds are mutually exclusive. So the worlds describe disjoint events and their probability sums up to 1. Given this information, what is the probability that φ_1 is true? As φ_1 is true in w_1 and w_2, we obtain $0.3 + 0.4 = 0.7$ as the probability that φ_1 is true. The probability of φ_2 being true is $0.3 + 0.2 = 0.5$.

[1] The author clearly prefers world w_2.

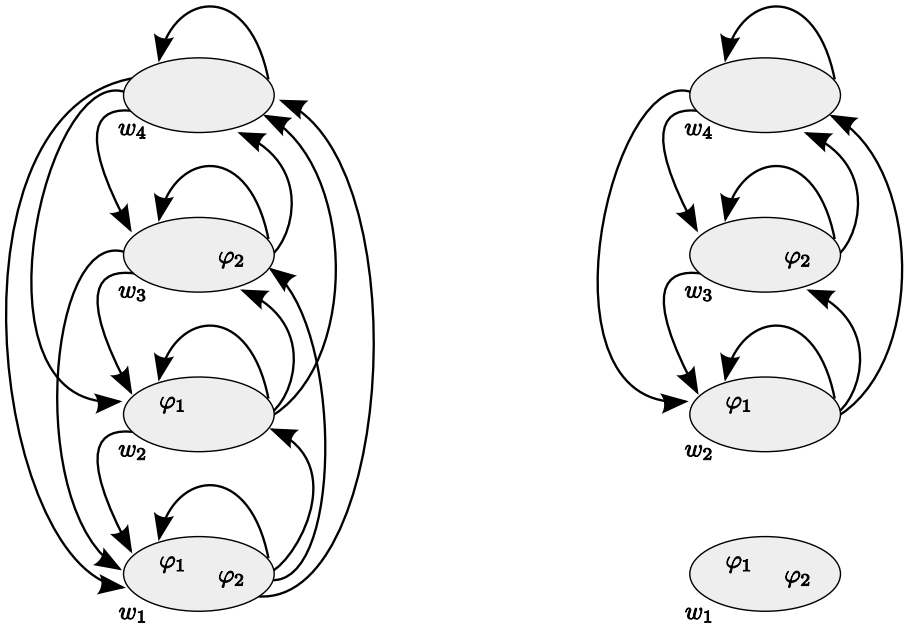


Figure 5.3: Worlds and accessibility relations of the football league example. The right hand side denotes the situation that the two clubs played against each other.

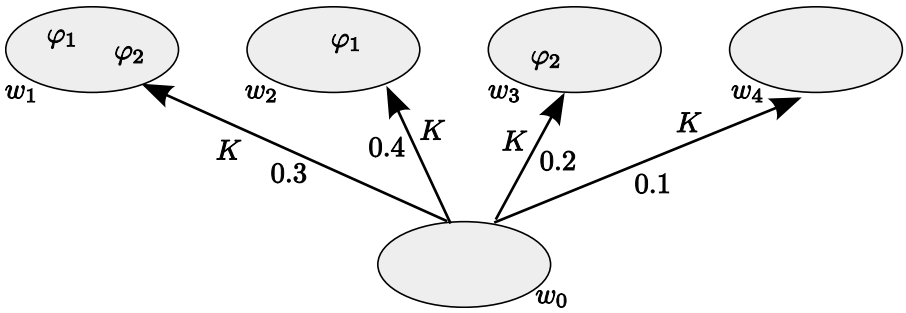


Figure 5.4: Possible worlds with probabilities

5.2.1.2 Four Truth Values

In the above example we dealt with the traditional two truth values – a proposition is either *true* or *false* in a world. In POLAR, we deal with four truth values – a proposition can either be *true*, *false*, *inconsistent* or *unknown* in a world. A proposition is *inconsistent*, if we have evidence that it is positive and negative. A proposition is *unknown* in case it is neither *true* nor *false*.

Using *inconsistent*, we can deal with contradicting assertions in annotations. For example, consider the POLAR program

```
d1[ *a1 *a2 ]
a1[ moon_made_of_cheese ]
a2[ !moon_made_of_cheese ]
```

Possible worlds	φ_1	φ_2
w_1	*true*	*true*
w_2	*true*	*false*
w_3	*true*	*inconsistent*
w_4	*true*	*unknown*
w_5	*false*	*true*
w_6	*false*	*false*
w_7	*false*	*inconsistent*
w_8	*false*	*unknown*
w_9	*inconsistent*	*true*
w_{10}	*inconsistent*	*false*
w_{11}	*inconsistent*	*inconsistent*
w_{12}	*inconsistent*	*unknown*
w_{13}	*unknown*	*true*
w_{14}	*unknown*	*false*
w_{15}	*unknown*	*inconsistent*
w_{16}	*unknown*	*unknown*

Table 5.1: 16 possible worlds with four truth values and two propositions

where annotation a1 states that the moon is made of cheese, and a2 contradicts. A reader of d1 has no knowledge about the question whether the moon is made of cheese when regarding only d1. But if the reader considers all contexts d1, a1 and a2, which is the augmented context d1(a1,a2), she gets inconsistent knowledge about the question whether the moon is made of cheese or not.

Furthermore, the truth value *unknown* can be used for an *open world assumption*; here, the absence of evidence that a proposition φ is *true* would not lead to the conclusion that it is *false*, like it would be the case when assuming a *closed world*, but it would simply be assigned the value *unknown*. As argued in (Rölleke, 1998, p. 24), the open world assumption is more reasonable in IR because term indexing is by nature incomplete – if a (human or automatic) indexer does not assign a term to a document, this does not necessarily mean that the document must not be indexed with that term. In that case, we would give this term the value *false* only if we were sure that the document cannot be indexed with it.

In the possible worlds model, if we have n truth values and m propositions, then there can be m^n possible worlds. In the above case of two propositions φ_1 and φ_2 and two truth values, we obtained four possible worlds. In the case of four truth values and two propositions, we get 16 possible worlds, as can be seen in Table 5.1.

Weight Lists and Truth Values In the description of the POLAR syntax in Figure 5.1 we introduced weight lists to convey four-valued probabilities. The four truth values defined in a weight list are all disjoint and sum up to 1. If only one weight is given, we take it as the probability that the corresponding proposition is *true*; the difference of this probability to 1 denotes the probability of the truth value *unknown*. For example, "0.7 φ" means that φ is *true* with 0.7 probability and *unknown* with a probability of 0.3. The second weight is the probability that the proposition is *false*. For instance, "0.7/0.1 φ" means that φ is *true* with 0.7 probability, *false* with 0.1 probability and *unknown* with a probability of 0.2 $(1 - (0.7 + 0.1))$. The third

weight gives the probability that the proposition is *inconsistent*. Therefore, "0.7/0.1/0.2 φ" means that φ is *true* with 0.7 probability, *false* with 0.1 probability and *inconsistent* with a probability of 0.2. As the difference to 1 is 0, the proposition is not *unknown* in this case.

Additionally, the expression "! φ" means "0/1 φ", and "φ" is a shortcut for "1 φ".

We will now continue with a discussion of POLAR's semantics for the basic knowledge modelling as shown in Figure 5.1. Afterwards we will discuss knowledge augmentation, i.e., the semantics of augmented contexts, and go on with rules and queries.

5.2.2 Basic Knowledge Modelling

To describe the semantics of basic POLAR programs, we apply the definitions and notions known from modal logics, using possible worlds and accessibility relations like explained above. From POOL we borrow the idea of regarding subparts as subcontexts. Additionally in POLAR, annotations, merged targets and fragments are seen as subcontexts w.r.t. the object they belong to, and so are references (or precisely: the referenced objects). For instance, the case that an annotation is accessed from a document is seen as accessing a subcontext (the annotation) from a supercontext (the document). The basic idea to describe our semantics is to regard each named context as an agent who can reach possible worlds. For example, the expression "`d1[...]`" would mean that an agent d1 reaches a set of possible worlds which is determined by the content of the context d1. Similarly, "`f1||...||`" describes an agent f1 which reaches certain possible worlds, "`t1<...>`" says that agent t1 reaches some worlds, "`*a1`" or "`@a1`" means agent a1 reaches certain worlds, and finally "`=>r1`" describes the worlds an agent r1 considers possible from the given context. If φ is a (non-probabilistic) proposition, then `d1[`φ`]` means that d1 believes or considers φ to be true. While Fagin et al. (1995) talk of *agents* which consider worlds possible or not, it is here *contexts* which access possible worlds. In the following, we say *contextagents* when we mean agents which reach and access worlds and which represent contexts, and stick to the notion of *contexts* if we mean a context in a POLAR expression. So if we say "from context d1, subcontext s1 is accessed", we mean "contextagent d1 can reach worlds from which contextagent s1 can reach other worlds" on the semantic level.

We will commence with the introduction of our probabilistic interpretation structure, which is based on Kripke structures used in modal logic. We present special propositions which encode the relation type (e.g., a subpart or content annotation relation). We then continue with the interpretation of basic POLAR programs, which includes the notion of validity, context-validity and context-seriality and the discussion of required constraints. All this leads to the definition of an interpretation structure M to be a model of a POLAR program P, which is denoted as $M \models P$. Making our way through it all, we illustrate many structures and give examples.

5.2.2.1 Interpretation Structure

We start with a formal definition of the interpretation structure underlying our further considerations. First, we repeat the definition of a probability space (see e.g. (Storch and Wiebe, 1989, p. 157ff.)).

Definition 4 (Probability space):
A *probability space* is a tuple (Ω, H, μ). Ω is a set of *basic events*, H is a so-called σ-algebra of Ω (a set of subsets of Ω closed under union and complement). $\mu : \Omega \rightarrow [0,1]$ is a *probability mass function*, which means: $\forall w \in \Omega$: $\mu(w) \geq 0$ and $\sum_{w \in \Omega} \mu(w) = 1$.

For the sake of simplicity, we define μ also as a function $\mu : H \rightarrow [0,1]$ which assigns a probability to each subset $A \in H$:

$$\forall A \in H : \mu(A) := \sum_{w \in A} \mu(w).$$

Our interpretation structure is the same as the one proposed in (Rölleke, 1998, p. 51) and Fagin and Halpern (1994) as a probabilistic extension of Kripke structures.

Definition 5 (Probabilistic interpretation structure):
A *probabilistic interpretation structure* M is a tuple

$$M = (W, \pi, R, P)$$

where W is a finite set of possible worlds and π is a *truth value assignment function* which assigns a truth value to each proposition in a given world $w \in W$:

$$\pi(w) : \Phi \longrightarrow \{true, false, inconsistent, unknown\}$$

with Φ as the set of propositions.

For a set of contextagents $\Lambda = \{a_1, \ldots, a_n\}$, $R = \{R_{a_1}, \ldots, R_{a_n}\}$ is a set of binary *accessibility relations* on W. For a tuple $(w, w') \in R_{a_i}$ we say that contextagent a_i *reaches* or *accesses* world w' from world w. To refer to the set of worlds which can be reached by a contextagent a_i from a world w through an accessibility relation, we define

$$R_{a_i}(w) := \{w' | (w, w') \in R_{a_i}\}.$$

P is a function that assigns a probability space $(W_{a_i,w}, H_{a_i,w}, \mu_{a_i,w})$ to each contextagent $a_i \in \Lambda$ and world w. $W_{a_i,w}$ is a subset of W and it is $W_{a_i,w} = R_{a_i}(w)$; $\mu_{a_i,w}(w')$ is thus the probability that a_i accesses a world $w' \in W_{a_i,w}$ from w.

In contrast to POOL, where we only deal with one subcontext and therefore one relation type, we cope with a range of relation and subcontext types in POLAR. But the set of accessibility relations R only contains information about the relations, but not their type. In order to distinguish between different relation types, we now introduce special propositions.

Definition 6 (Special propositions for relation types):
In order to distinguish between the relation types subpart, merged annotation target, fragment, (positive and negative) content annotation, (positive and negative) meta annotation and reference, we introduce the following special propositions:

_subpart(sp) This proposition is used if we want to denote that subcontext sp is reached via a subpart relation.

_mtarget(mt) This proposition is used if subcontext mt is a merged annotation target.

_fragment(fr) This proposition is used if subcontext fr is a fragment.

_canno(ca) This proposition is true in case the subcontext ca is a content annotation.

_negcanno(ca) means that subcontext ca is a negative content annotation.

_manno(ma) stands for the case that the subcontext ma is a meta level annotation.

_negmanno(ma) says that the subcontext ma is a negative meta level annotation.

_reference(rf) denotes that subcontext rf is a referenced object.

Each of these propositions underlies a closed world assumption, and we only allow the two truth values *true* and *false* for them. If, for example, `_subpart(sp)` is not *true*, this automatically means that `_subpart(sp)` is *false*.

One constraint needs to be defined regarding the uniqueness and consistency of the access relation type. We disallow that an agent can reach a subcontext from a given context through more than one relation type, e.g. something like

```
d[ *a @a ]
d[ *a a[] ]
```

is forbidden.

Constraint 7 (Relation type consistency): In all worlds w reachable from a world w_0 by a contextagent d, if there is a relation type proposition regarding another context c, then c's relation type must be consistent. This means that there exists no other sister world w' reachable from a world w_0 by a contextagent d so that c has another relation type in w', but in each of these sister worlds the same relation type for c is true as in w. More formally, $\forall(w, w' \in R_d(w_o), c \in \Lambda)$:

$$\begin{aligned}\pi(w)(\text{\_subpart(c)}) = \mathit{true} \Longrightarrow \\ \pi(w')(\text{\_mtarget(c)}) = \mathit{false} \wedge \pi(w')(\text{\_canno(c)}) = \mathit{false} \wedge \\ \pi(w')(\text{\_negcanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_manno(c)}) = \mathit{false} \wedge \\ \pi(w')(\text{\_negmanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_reference(c)}) = \mathit{false} \wedge \\ \pi(w')(\text{\_fragment(c)}) = \mathit{false} \wedge \pi(w')(\text{\_subpart(c)}) = \mathit{true}\end{aligned}$$

$$\pi(w)(\text{\_mtarget(c)}) = \mathit{true} \Longrightarrow$$
$$\pi(w')(\text{\_mtarget(c)}) = \mathit{true} \wedge \pi(w')(\text{\_canno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negcanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_manno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negmanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_reference(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_fragment(c)}) = \mathit{false} \wedge \pi(w')(\text{\_subpart(c)}) = \mathit{false}$$

$$\pi(w)(\text{\_canno(c)}) = \mathit{true} \Longrightarrow$$
$$\pi(w')(\text{\_mtarget(c)}) = \mathit{false} \wedge \pi(w')(\text{\_canno(c)}) = \mathit{true} \wedge$$
$$\pi(w')(\text{\_negcanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_manno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negmanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_reference(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_fragment(c)}) = \mathit{false} \wedge \pi(w')(\text{\_subpart(c)}) = \mathit{false}$$

$$\pi(w)(\text{\_negcanno(c)}) = \mathit{true} \Longrightarrow$$
$$\pi(w')(\text{\_mtarget(c)}) = \mathit{false} \wedge \pi(w')(\text{\_canno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negcanno(c)}) = \mathit{true} \wedge \pi(w')(\text{\_manno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negmanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_reference(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_fragment(c)}) = \mathit{false} \wedge \pi(w')(\text{\_subpart(c)}) = \mathit{false}$$

$$\pi(w)(\text{\_manno(c)}) = \mathit{true} \Longrightarrow$$
$$\pi(w')(\text{\_mtarget(c)}) = \mathit{false} \wedge \pi(w')(\text{\_canno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negcanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_manno(c)}) = \mathit{true} \wedge$$
$$\pi(w')(\text{\_negmanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_reference(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_fragment(c)}) = \mathit{false} \wedge \pi(w')(\text{\_subpart(c)}) = \mathit{false}$$

$$\pi(w)(\text{\_negmanno(c)}) = \mathit{true} \Longrightarrow$$
$$\pi(w')(\text{\_mtarget(c)}) = \mathit{false} \wedge \pi(w')(\text{\_canno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negcanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_manno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negmanno(c)}) = \mathit{true} \wedge \pi(w')(\text{\_reference(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_fragment(c)}) = \mathit{false} \wedge \pi(w')(\text{\_subpart(c)}) = \mathit{false}$$

$$\pi(w)(\text{\_fragment(c)}) = \mathit{true} \Longrightarrow$$
$$\pi(w')(\text{\_mtarget(c)}) = \mathit{false} \wedge \pi(w')(\text{\_canno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negcanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_manno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negmanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_reference(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_fragment(c)}) = \mathit{true} \wedge \pi(w')(\text{\_subpart(c)}) = \mathit{false}$$

$$\pi(w)(\text{\_reference(c)}) = \mathit{true} \Longrightarrow$$
$$\pi(w')(\text{\_mtarget(c)}) = \mathit{false} \wedge \pi(w')(\text{\_canno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negcanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_manno(c)}) = \mathit{false} \wedge$$
$$\pi(w')(\text{\_negmanno(c)}) = \mathit{false} \wedge \pi(w')(\text{\_reference(c)}) = \mathit{true} \wedge$$
$$\pi(w')(\text{\_fragment(c)}) = \mathit{false} \wedge \pi(w')(\text{\_subpart(c)}) = \mathit{false}$$

If c is declared as being a subpart in w, then it must not be declared as something else in a sister world w' of w or in w itself. The same holds for the other relation types.

□

5.2.2.2 Interpretation of Basic POLAR Programs

Our goal is to define a suitable interpretation structure for the basic POLAR syntax as described in Figure 5.1. For this we have to show that a probabilistic interpretation structure M as defined in Definition 5 is a model of the given POLAR program. To do so, we use the notation "$(M, w) \models \mathit{expr}$" to denote that a POLAR expression *expr* is true with respect to the interpretation structure M and a world w.

Interpretation of Probabilistic Propositions For a proposition φ, $(M, w) \models_t \varphi$ means that φ is *true* with respect to interpretation M and world w. Similarly, we define $\models_f$ for *false*, $\models_i$ for *inconsistent* and $\models_u$ for *undefined*. We then obtain the same interpretation of propositions as in Rölleke (1998):

Definition 7 (Interpretation of propositions):
The interpretation of propositions in a world w is defined using the truth value assignment function in that world:

$$\begin{aligned}
(M, w) \models_t \varphi &\iff \pi(w)(\varphi) = \mathit{true} \\
(M, w) \models_f \varphi &\iff \pi(w)(\varphi) = \mathit{false} \\
(M, w) \models_i \varphi &\iff \pi(w)(\varphi) = \mathit{inconsistent} \\
(M, w) \models_u \varphi &\iff \pi(w)(\varphi) = \mathit{unknown}
\end{aligned}$$

Based on this, we can now define the interpretation of probabilistic propositions within the different kinds of POLAR contexts. Such propositions are, for example, "`d[ 0.2/0.7 foo ]`" for (sub)parts, but also "`f|| 0.3/0.2/0.2 bar ||`" for fragments and "`m< 0.3 foobar >`" for merged annotation targets. Let p_t be the probability that the truth value *true* is assigned (similarly for p_f, p_i and p_u). We then define the interpretation of probabilistic propositions in contexts as follows:

Definition 8 (Interpretation of prob. propositions in contexts):
For subpart contexts, we define

$$\begin{aligned}
(M,w) \models\ & \mathrm{d}[p_t/p_f/p_i/p_u\ \varphi] \Longleftrightarrow \\
& p_t = \mu_{d,w}(\{w' | (M,w) \models_t _\mathrm{subpart(d)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_t \varphi\}) \quad \text{and} \\
& p_f = \mu_{d,w}(\{w' | (M,w) \models_t _\mathrm{subpart(d)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_f \varphi\}) \quad \text{and} \\
& p_i = \mu_{d,w}(\{w' | (M,w) \models_t _\mathrm{subpart(d)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_i \varphi\}) \quad \text{and} \\
& p_u = \mu_{d,w}(\{w' | (M,w) \models_t _\mathrm{subpart(d)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_u \varphi\})
\end{aligned}$$

The probabilities p_t, p_f, p_i and p_u are the sum over the probabilities of the possible worlds, reachable by a subpart relation, where the proposition is assigned the corresponding truth value.

In a similar manner, we define the interpretation of probabilistic propositions in merged annotation targets:

$$\begin{aligned}
(M,w) \models\ & \mathrm{m}\!<\!p_t/p_f/p_i/p_u\ \varphi\!> \Longleftrightarrow \\
& p_t = \mu_{m,w}(\{w' | (M,w) \models_t _\mathrm{mtarget(m)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_t \varphi\}) \quad \text{and} \\
& p_f = \mu_{m,w}(\{w' | (M,w) \models_t _\mathrm{mtarget(m)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_f \varphi\}) \quad \text{and} \\
& p_i = \mu_{m,w}(\{w' | (M,w) \models_t _\mathrm{mtarget(m)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_i \varphi\}) \quad \text{and} \\
& p_u = \mu_{m,w}(\{w' | (M,w) \models_t _\mathrm{mtarget(m)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_u \varphi\})
\end{aligned}$$

This time, we only consider worlds reached via a merged target relation.

For propositions in a fragment we define

$$\begin{aligned}
(M,w) \models\ & \mathrm{fr}||p_t/p_f/p_i/p_u\ \varphi|| \Longleftrightarrow \\
& p_t = \mu_{fr,w}(\{w' | (M,w) \models_t _\mathrm{fragment(fr)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_t \varphi\}) \quad \text{and} \\
& p_f = \mu_{fr,w}(\{w' | (M,w) \models_t _\mathrm{fragment(fr)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_f \varphi\}) \quad \text{and} \\
& p_i = \mu_{fr,w}(\{w' | (M,w) \models_t _\mathrm{fragment(fr)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_i \varphi\}) \quad \text{and} \\
& p_u = \mu_{fr,w}(\{w' | (M,w) \models_t _\mathrm{fragment(fr)} \wedge \\
& \qquad\qquad w' \in R_d(w) \wedge (M,w') \models_u \varphi\})
\end{aligned}$$

In the definition above, we used the phrase "reachable by a subpart relation". If we say that a world w' is reachable by a subpart (or any other of the relation types) relation from another world w by an agent a, we precisely mean: $(M, w) \models_t$ _subpart(a) and $w' \in R_a(w)$ (exchange "_subpart(a)" with an appropriate special proposition in case of other relation types).

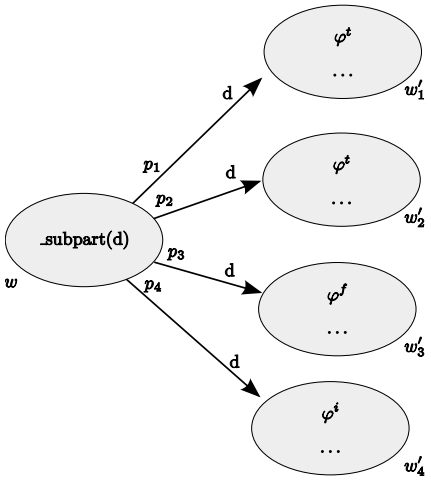


Figure 5.5: Example interpretation of "`d[`$p_t/p_f/p_i/p_u$ φ`]`". φ^t means "φ is *true*". Analogously for φ^f and φ^i.

Example 4 (Probabilistic propositions in contexts): Fig. 5.5 shows an example interpretation of probabilistic propositions, here in a subpart context. φ is *true* in the worlds w'_1 and w'_2, *false* in w'_3 and *inconsistent* in w'_4. According to the definition, the probabilities in this example calculate as follows: $p_t = p_1 + p_2, p_f = p_3$ and $p_i = p_4$. We can think of similar examples for merged targets and fragments. □

Global Database Context We regard all propositions as being embedded in a context. This leads to one problem. Consider the POLAR program

```
document(d1)
```

which consists of the proposition that d1 is a document. In which context is this proposition made? Propositions like the one above are defined outside any document or annotation, so we regard them as a kind of global database knowledge. We therefore define a special database context which we call *db-context* and which is accessed by the agent db. The above proposition is thus a shortcut for `db[document(d1)]`.

Probabilistic Access We defined the semantics for propositions in a context. But what does it mean in our model to access a subcontext with a certain probability (which we called the access probability)? This is subject to discussion now.

Syntactically, probabilistic access is expressed in different ways. Subparts, fragments and merged targets have exactly one supercontext they are accessed from, so we could encode them and their content in expressions like "`d[0.4 s[...]]`", '`d[0.1 f||...||]`" or "`d[0.6 m<...>]`". With annotations and references, this is a different story, since theoretically, an annotation can annotate more than one target respectively an object may be

referenced by more than one annotation. This is the reason why the access to these subcontexts is expressed with a different syntax like "`d[0.4 *a]`", "`d[0.5 @a]`" or "`d[0.6 =>o]`", which may remind one of a more pointer-oriented syntax as it is found in certain programming languages like C.

We continue with a definition of probabilistic access in annotations and references and then present merged targets, fragments and subparts.

Definition 9 (Probabilistic access (annotations and references)):
Let p be the access probability. For content and meta annorefs and for references in subparts we define

$$\begin{aligned}
(M,w) \models \mathrm{d}[p\ {*}\mathrm{a}] &\iff p = \mu_{d,w}(\{w' | (M,w) \models_t _\mathrm{subpart(d)} \wedge w' \in R_d(w) \wedge \\
&\qquad (M,w') \models_t _\mathrm{canno(a)} \wedge R_a(w') \neq \emptyset\}) \\
(M,w) \models \mathrm{d}[p\ {-}{*}\mathrm{a}] &\iff p = \mu_{d,w}(\{w' | (M,w) \models_t _\mathrm{subpart(d)} \wedge w' \in R_d(w) \wedge \\
&\qquad (M,w') \models_t _\mathrm{negcanno(a)} \wedge R_a(w') \neq \emptyset\}) \\
(M,w) \models \mathrm{d}[p\ @\mathrm{a}] &\iff p = \mu_{d,w}(\{w' | (M,w) \models_t _\mathrm{subpart(d)} \wedge w' \in R_d(w) \wedge \\
&\qquad (M,w') \models_t _\mathrm{manno(a)} \wedge R_a(w') \neq \emptyset\}) \\
(M,w) \models \mathrm{d}[p\ {-}@\mathrm{a}] &\iff p = \mu_{d,w}(\{w' | (M,w) \models_t _\mathrm{subpart(d)} \wedge w' \in R_d(w) \wedge \\
&\qquad (M,w') \models_t _\mathrm{negmanno(a)} \wedge R_a(w') \neq \emptyset\}) \\
(M,w) \models \mathrm{d}[p\ {=>}\ \mathrm{o}] &\iff p = \mu_{d,w}(\{w' | (M,w) \models_t _\mathrm{subpart(d)} \wedge w' \in R_d(w) \wedge \\
&\qquad (M,w') \models_t _\mathrm{reference(a)} \wedge R_o(w') \neq \emptyset\}).
\end{aligned}$$

From a context d, content annotation a is accessed with probability p iff p is the sum of the probabilities of all worlds that

- contextagent d reaches from w through a subpart relation and
- from which contextagent a reaches at least one world through a content annotation relation.

The access of negative and meta annotations, and references is defined analogously.

A similar definition applies for fragments:

$$\begin{aligned}
(M,w) \models \mathrm{fr}||p\ {*}\mathrm{a}|| &\iff p = \mu_{fr,w}(\{w' | (M,w) \models_t _\mathrm{fragment(fr)} \wedge w' \in R_{fr}(w) \wedge \\
&\qquad (M,w') \models_t _\mathrm{canno(a)} \wedge R_a(w') \neq \emptyset\}) \\
(M,w) \models \mathrm{fr}||p\ {-}{*}\mathrm{a}|| &\iff p = \mu_{fr,w}(\{w' | (M,w) \models_t _\mathrm{fragment(fr)} \wedge w' \in R_{fr}(w) \wedge \\
&\qquad (M,w') \models_t _\mathrm{negcanno(a)} \wedge R_a(w') \neq \emptyset\}) \\
(M,w) \models \mathrm{fr}||p\ @\mathrm{a}|| &\iff p = \mu_{fr,w}(\{w' | (M,w) \models_t _\mathrm{fragment(fr)} \wedge w' \in R_{fr}(w) \wedge \\
&\qquad (M,w') \models_t _\mathrm{manno(a)} \wedge R_a(w') \neq \emptyset\}). \\
(M,w) \models \mathrm{fr}||p\ {-}@\mathrm{a}|| &\iff p = \mu_{fr,w}(\{w' | (M,w) \models_t _\mathrm{fragment(fr)} \wedge w' \in R_{fr}(w) \wedge \\
&\qquad (M,w') \models_t _\mathrm{negmanno(a)} \wedge R_a(w') \neq \emptyset\}).
\end{aligned}$$

Note that fr is only defined as a subcontext of another context, as fragments are always parts of documents. Nevertheless, we need the above definition later. Access to annotations or references is not defined for merged annotation targets, so we do not need a definition for this case.

We now define the probabilistic access to merged annotation targets, subparts and fragments. We start with merged annotation targets and fragments.

Definition 10 (Probabilistic access (merged targets and fragments)):
Factlists are used as the content of both merged targets and fragments. Factlists consist of facts, which are propositions with a weightlist. Let $factlist^*$ be the set of facts in a factlist. We then define:

$$(M,w) \models \text{d}[p\ \text{m<factlist>}] \Longleftrightarrow$$
$$p = \mu_{d,w}(\{w' | (M,w) \models_t \text{\_subpart(d)} \wedge w' \in R_d(w) \wedge$$
$$\forall fact \in factlist^* : (M,w') \models \text{m<}fact\text{>} \wedge$$
$$(M,w') \models_t \text{\_mtarget(m)} \wedge R_m(w') \neq \emptyset\})$$

An interpretation structure M is a model of d[p m<factlist>] with respect to the world w iff p is the probability of all worlds w' which

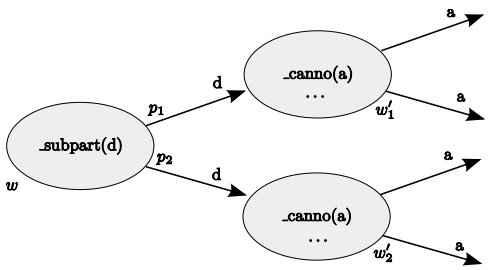
- can be accessed by d through a subpart relation from w and
- for which (M,w') is a model of each fact in the merged annotation target m (see Def. 8) and
- from which contextagent m can reach other worlds through a merged target relation, which means that d must know that m is a merged target and $R_m(w')$ is not empty.

With a similar argumentation we define for the access to fragments containing a fact list:

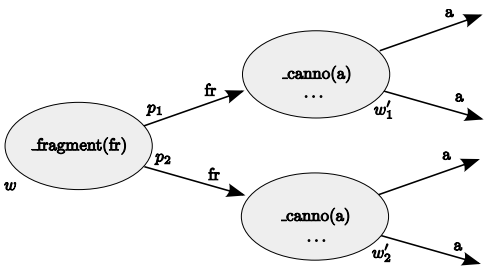
$$(M,w) \models \text{d}[p\ \text{fr}||\text{factlist annoreflist}||] \Longleftrightarrow$$
$$p = \mu_{d,w}(\{w' | (M,w) \models_t \text{\_subpart(d)} \wedge w' \in R_d(w) \wedge$$
$$\forall fact \in factlist^* : (M,w') \models \text{fr}||fact|| \wedge$$
$$\forall annoref \in annoreflist^* : (M,w') \models \text{fr}||annoref|| \wedge$$
$$(M,w') \models_t \text{\_fragment(fr)} \wedge R_{fr}(w') \neq \emptyset\})$$

Besides a fact list, fragments can also contain up to one annoref. This annoref is stored in an annoreflist, which can contain 1 or 0 annorefs. Similar to $factlist^*$, $annoreflist^*$ is the set of annorefs in the annoreflist. Annorefs are either (positive or negative) content or meta annotation references with an access probability. (M,w) is a model of d[p fr||factlist annoref||] iff p is the probability of all worlds w' which can be accessed by d from w and for which (M,w') is a model of each fact in the fragment fr and of the annoref in fr (see Def. 9).

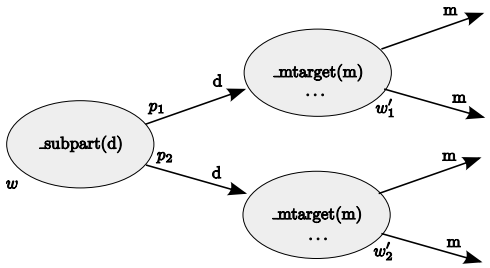
The content of a subpart is a special contextprogram, which can contain subparts again. So our definition of probabilistic access for subparts is recursive.



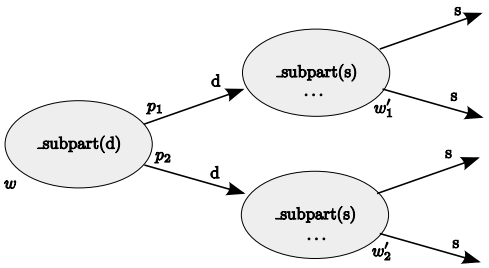
(a) Subpart to content annotation: `d[p *a]`



(b) Fragment to content annotation: `fr||p *a||`



(c) Subpart to merged target: `d[p m<...>]`



(d) Subpart to subpart: `d[p s[...]]`

Figure 5.6: Examples of probabilistic access. $p = p_1 + p_2$ in all cases.

Definition 11 (Probabilistic access (subparts)):
A contextprogram contains contexclauses, which can be facts, annorefs, references, fragments or subparts. Let $contextprogram^*$ be the set of contextclauses appearing in the contextprogram. We define:

$$
\begin{aligned}
&(M, w) \models \mathrm{d}[p\ \mathrm{s}[\mathrm{contextprogram}]] \iff \\
&\quad p = \mu_{d,w}(\{w' | (M, w) \models_t \_\mathrm{subpart(d)} \wedge w' \in R_d(w) \wedge \\
&\qquad\qquad \forall contextclause \in contextprogram^* : (M, w') \models \mathrm{s}[contextclause] \wedge \\
&\qquad\qquad (M, w') \models_t \_\mathrm{subpart(s)} \wedge R_s(w') \neq \emptyset\})
\end{aligned}
$$

Example 5 (Probabilistic access): Figure 5.6 shows four examples of probabilistic access and snippets of their corresponding interpretation structure. □

Inconsistent Belief In the definitions 9 to 11, when calculating μ, we only considered worlds which can reach other worlds through an appropriate relation (for example, in Def. 11 we required $R_s(w') \neq \emptyset$). What happens in the case that we reach a world w from which a contextagent d does not reach any other world, so that $R_d(w) = \emptyset$? In (Rölleke, 1998, p. 37), this case is called *inconsistent belief* – in a world where an agent does not reach any other world (including the world the agent currently is in), it believes in every possible truth value

for propositions and every possible access[2]. So in a world w with $R_d(w) = \emptyset$, it is $(M, w) \models \mathrm{d}[p\,\varphi]$ and also $(M, w) \models \mathrm{d}[q\,\varphi]$, with $p \neq q$. Inconsistent belief is useful for defining valid knowledge, i.e. knowledge which is true in every world.

Validity of POLAR Programs Following Fagin et al. (1995), we define the validity of expressions. An expression is valid in an interpretation structure M if it is true w.r.t. all M's worlds.

Definition 12 (Validity):
For each expression *expr*, we say that *expr* is *valid* in M if $\forall w \in W : (M, w) \models expr$. In this case we write $M \models expr$.

Based on the definitions above, we formulate our first requirement for a structure M to be model of a POLAR program P, which says that db[P] must be true in all worlds w.r.t. M:

Constraint 8 (Validity of POLAR programs): If M is a model of a POLAR program P, denoted as $M \models P$, then db[P] must be valid w.r.t. M:

$$M \models P \Longrightarrow M \models \mathrm{db}[P].$$

db is the global database context defined above. □

Note that the validity of db[P] is possible due to inconsistent belief – in worlds w with $R_{db}(w) = \emptyset$, the contextagent db considers everything possible.

Before discussing further requirements, we give another example.

Example 6 (Interpretation structures): Consider the following POLAR program P:

```
d1[foo *a1]
a1[ !foo]
foo
```

Now consider 4 possible worlds $W = \{w_{root}, w_0, w_1, w_2\}$ with

$$\begin{aligned} \pi(w_0)(\_\mathrm{subpart(d1)}) &= \mathit{true} \\ \pi(w_0)(\_\mathrm{subpart(a1)}) &= \mathit{true} \\ \pi(w_0)(\mathrm{foo}) &= \mathit{true} \\ \pi(w_1)(\_\mathrm{canno(a1)}) &= \mathit{true} \\ \pi(w_1)(\mathrm{foo}) &= \mathit{true} \\ \pi(w_2)(\mathrm{foo}) &= \mathit{false} \end{aligned}$$

[2]Inconsistent belief is not to be mixed up with the truth value *inconsistent*

(in w_{root}, every proposition is *unknown*) and the following accessibility relations

$$\begin{aligned}
R_{db}^{(1)} &= \{(w_{root}, w_0)\} \\
R_{d1}^{(1)} &= \{(w_0, w_1)\} \\
R_{a1}^{(1)} &= \{(w_0, w_2), (w_1, w_2)\} \\
R_{db}^{(2)} &= \{(w_{root}, w_0), (w_2, w_1)\} \\
R_{d1}^{(2)} &= \{(w_0, w_1)\} \\
R_{a1}^{(2)} &= \{(w_0, w_2), (w_1, w_2)\} \\
R_{db}^{(3)} &= \{(w_{root}, w_0)\} \\
R_{d1}^{(3)} &= \emptyset \\
R_{a1}^{(3)} &= \{(w_0, w_2)\}
\end{aligned}$$

We define 3 interpretation structures M_1, M_2 and M_3 as:

$$\begin{aligned}
M_1 &= (W, \pi, \{R_{db}^{(1)}, R_{d1}^{(1)}, R_{a1}^{(1)}\}, Pr) \\
M_2 &= (W, \pi, \{R_{db}^{(2)}, R_{d1}^{(2)}, R_{a1}^{(2)}\}, Pr) \\
M_3 &= (W, \pi, \{R_{db}^{(3)}, R_{d1}^{(3)}, R_{a1}^{(3)}\}, Pr)
\end{aligned}$$

(Pr assigns the probability 1 to each relation). The 3 interpretation structures are depicted in Fig. 5.7. □

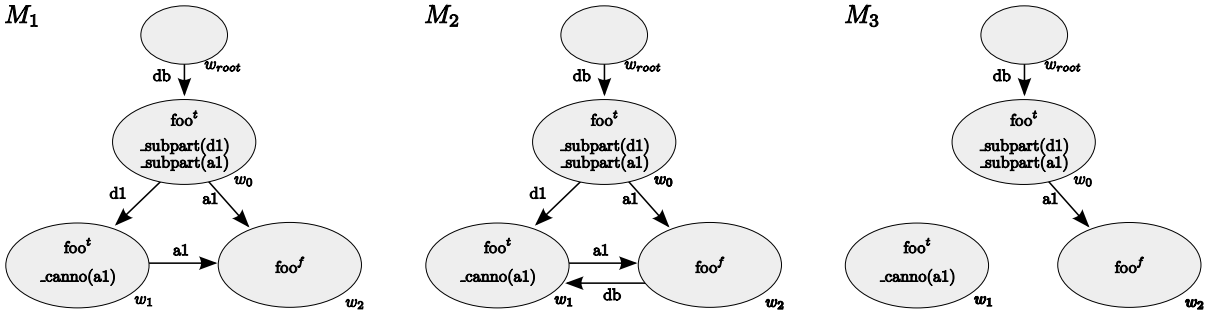


Figure 5.7: Three interpretation structures

In which of the structures is the above POLAR program valid? It is valid in M_1, since $(M, w_{root}) \models \text{db}[P]$ and in w_0, w_1 and w_2, contextagent db considers everything possible (due to inconsistent belief), so our program is true here as well. This means that P is valid in M_1. But P is not valid in M_2. The problem here is the world w_2. Here, db[foo] and db[_canno(a1)] are true, but this also means that db[P] is not true in this world, making P invalid w.r.t. M_2. An interesting case is M_3 because surprisingly, db[P] is valid here. We can also see that $(M, w_0) \models$ d1[foo *a1] due to the inconsistent belief contextagent d1 has in w_0. Can we conclude that M_3 is a good candidate for being a model of our program P? If we argue

like that, then even a structure M with empty accessibility relations ($R_a = \emptyset$ $\forall$contextagent a) would be a candidate for a model, since P is valid here as well! It is easy to see that this would lead to very absurd interpretation structures which are useless in the end, so we have to formulate another constraint, besides validity, for models of POLAR programs.

Constraint 9 (Existence of accessibility relations): For each contextagent $a \in \Lambda$ appearing in a POLAR program (including the db-agent), $R_a \neq \emptyset$ holds. □

This constraint excludes M_3 from the list of candidate models for the POLAR program.

In Example 6, we introduced a world w_{root}. This world is needed as a starting point from which the db-agent reaches possible worlds. We will thus use this world in a similar way in our further considerations. Per default, all propositions are *unknown* in w_{root}.

While inconsistent belief helped us in the example above to make sure that `db[P]` is valid w.r.t. an interpretation structure M ($M \models \mathrm{db}[P]$), inconsistent belief has some side effect which needs to be handled as well. In particular, we are talking about context-validity and context-seriality.

Context-Validity Consider our POLAR program in Example 6 again. If we take the interpretation structure M_1, then, e.g., d1[foo] is valid (in w_0 d1 reaches a world where foo is *true*, in any other world d1 considers everything possible). This means that for example in w_o, a1 reaches worlds in which d1[foo] is true, so $(M_1, w_0) \models$ a1[d1[foo]], which according to Definition 11 implies:

$$\begin{aligned} 1 = \mu_{a1,w_0}(\{w' | (M, w_0) \models_t \_\mathrm{subpart(d)} \wedge w' \in R_{a1}(w_0) \wedge \\ (M, w') \models \mathrm{d1[foo]} \wedge \\ (M, w') \models_t \_\mathrm{subpart(d1)} \wedge R_{d1}(w') \neq \emptyset\}) \end{aligned}$$

which says "*if* there are worlds w' according to the conditions above, then the sum of their probabilities must be 1". Since there are no such worlds ($(M, w2) \not\models_t$ _subpart(d1) and $R_{d1}(w_2) = \emptyset$), this implication still holds. a1[d1[foo]] is even valid in M, as we can easily check. This contradicts the logical structure of subcontexts we find in P, where a1 is a subcontext of d1, but not the other way round. Following the discussion in (Rölleke, 1998, pp. 36), we introduce the notion of *context-validity*. But before we have to (informally) define a function *sub* which yields the set of direct subcontexts for a supercontext. For example, consider the program

```
d1[ s1[] s2[s21[]] f1 ||*a1||]
a1[ a12[]]
```

Here, $sub(\mathrm{db}) = \{\mathrm{d1}, \mathrm{a1}\}$ and $sub(\mathrm{d1}) = \{\mathrm{s1}, \mathrm{s2}, \mathrm{f1}, \mathrm{a1}\}$ (due to fragment permeability, see below). Furthermore, $sub(\mathrm{a1}) = \{\mathrm{a12}\}$, $sub(\mathrm{f1}) = \{\mathrm{a1}\}$ and $sub(\mathrm{s2}) = \{\mathrm{s21}\}$.

Definition 13 (Context-validity):
With the help of the *sub* function, we define the *context-validity* $\models_C$:

$$
\begin{aligned}
M \models_C \text{d}[p\ \text{s}[p^t/p^f/p^i/p^u\varphi]] &\iff M \models \text{d}[p\ \text{s}[p^t/p^f/p^i/p^u\varphi]] \text{ and } s \in sub(d)\\
M \models_C \text{d}[p\ \text{m}\!<\!p^t/p^f/p^i/p^u\varphi\!>] &\iff M \models \text{d}[p\ \text{m}\!<\!p^t/p^f/p^i/p^u\varphi\!>] \text{ and } m \in sub(d)\\
M \models_C \text{d}[p\ \text{f}||p^t/p^f/p^i/p^u\varphi||] &\iff M \models \text{d}[p\ \text{f}||p^t/p^f/p^i/p^u\varphi||] \text{ and } f \in sub(d)\\
M \models_C \text{d}[p\ \text{*a}] &\iff M \models \text{d}[p\ \text{*a}] \text{ and } a \in sub(d)\\
M \models_C \text{d}[p\ \text{-*a}] &\iff M \models \text{d}[p\ \text{-*a}] \text{ and } a \in sub(d)\\
M \models_C \text{d}[p\ \text{@a}] &\iff M \models \text{d}[p\ \text{@a}] \text{ and } a \in sub(d)\\
M \models_C \text{d}[p\ \text{-@a}] &\iff M \models \text{d}[p\ \text{-@a}] \text{ and } a \in sub(d)\\
M \models_C \text{d}[p\ \text{=>r}] &\iff M \models \text{d}||p\ \text{=>r}|| \text{ and } r \in sub(d)\\
M \models_C \text{f}||p\ \text{*a}|| &\iff M \models \text{f}||p\ \text{*a}|| \text{ and } a \in sub(f)\\
M \models_C \text{f}||p\ \text{-*a}|| &\iff M \models \text{f}||p\ \text{-*a}|| \text{ and } a \in sub(f)\\
M \models_C \text{f}||p\ \text{@a}|| &\iff M \models \text{f}||p\ \text{@a}|| \text{ and } a \in sub(f)\\
M \models_C \text{f}||p\ \text{-@a}|| &\iff M \models \text{f}||p\ \text{-@a}|| \text{ and } a \in sub(f)
\end{aligned}
$$

An expression is context-valid if it is valid and follows the subcontext relations.

According to this definition, d1[a1[!foo]] is context-valid in M_1 in Example 6, but a1[d1[foo]] is not. So $M \models$ a1[d1[foo]], but $M \not\models_C$ a1[d1[foo]]

Context-Seriality We refer to Example 6 again. Let us assume an interpretation structure $M_4 = (W, \pi, \{R^{(4)}_{db}, R^{(4)}_{d1}, R^{(4)}_{d1}\})$ with $R^{(4)}_{db} = \{(w_{root}, w_0)\}, R^{(4)}_{d1} = \{(w_0, w_1)\}, R^{(4)}_{a1} = \{(w_0, w_2)\}$. This structure is shown in Figure 5.8 on the left side. From our considerations so far we know that

- Constraint 9 is satisfied,
- $M_4 \models P$ and also
- $M \models_C$ d1[a1[!foo]].

But there is still something wrong with M_4, because although $M \models_C$ d1[a1[!foo]], this is based on inconsistent belief since from w_1, a1 does not reach any other world (and in particular not w_2). The structure of our POLAR program is not reflected properly by M_4 – there should be at least one world which is reachable by a1 from at least one world reachable by d1, so $R_{a1}(w_1)$ should not be empty. This is ensured by the concept of context-seriality:

Definition 14 (Context-seriality):
An accessibility relation of a subcontext s is *context-serial* with respect to a supercontext d and a relation type t if and only if: for each world w from which d can reach other worlds,

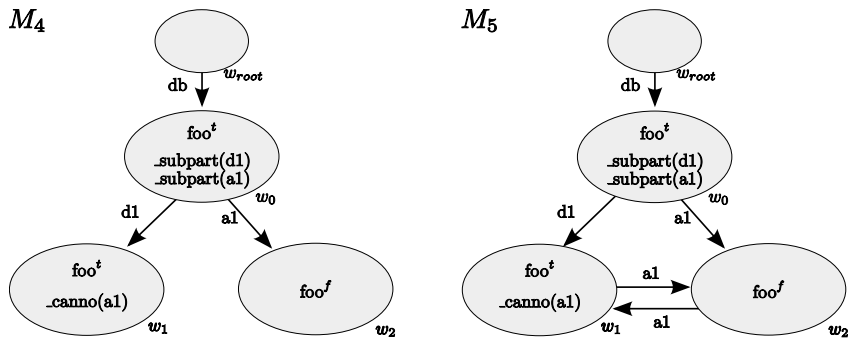


Figure 5.8: Two other interpretation structures

there exists a world w' reachable by d from w in which the type of s w.r.t. d is known and from which s reaches at least one other world. Formally:

R_s is context-serial w.r.t. d and type $t \iff$
$$(\forall w \in W : R_d(w) \neq \emptyset \Rightarrow \exists w' \in R_d(w) : (M, w') \models_t t(s) \wedge R_s(w') \neq \emptyset)$$

$t(s)$ stands for the subcontext type and is substituted by the corresponding special proposition. For instance, if s is a content annotation, then $t(s) =$ _contentanno(s).

From the definition of context-seriality, we arrive at another constraint:

Constraint 10 (Context-seriality): If $s \in sub(d)$ then R_s must be context-serial with respect to the supercontext d and its type t. □

Constraint 7 on page 98 prohibits that the knowledge about the relation type is inconsistent. Constraint 9 ensures that there exists a world w so that $R_a(w) \neq \emptyset$, so we have a starting point for seriality for every contextagent. Note that context-seriality ensures the conditions needed in Definition 9, 10 and 11. Referring back to Example 6, the context-seriality constraint is obeyed in M_1 and even M_3, but not in M_2 (d1 is a subcontext of db, but $R_{d1}(w_1) = \emptyset$) and M_4 (as discussed). Context-seriality is also a crucial property when it comes to knowledge augmentation later.

Valid Knowledge of Contextagents Again we turn to Example 6, but now we regard the structure M_5 which can be seen on the right side of Fig. 5.8. All constraints are respected, but we are still not satisfied. What is disturbing us is the edge from w_2 to w_1 labelled with a1. Because of this edge, a1 has different knowledge in w_2 than it has in w_0 and w_1. It is clear that the knowledge a_1 has in w_0 and w_1 reflects our program, whereas the knowledge in w_2 does not – it even contradicts what a1 is supposed to know! We can prohibit such a situation by formulating the following constraint:

Constraint 11 (Valid Knowledge of Contextagents): The knowledge a contextagent has is valid. □

The consequence of this constraint is, for our example:

$$(M, w) \models \text{a1[!p]} \Longrightarrow M \models \text{a1[!p]}$$

M_5 violates this constraint due to the edge between w_2 and w_1.

Fragment Permeability Fragment access has some peculiarity. We say that fragments are *permeable w.r.t. annotation*, which means that every annotation annotating a fragment also annotates the object the fragment belongs to. Therefore the expressions `d1[f1||... *a||]` and `d1[*a f1||... *a||]` must be semantically equivalent. This leads us to another constraint:

Constraint 12 (Fragment Permeability): If f is a fragment of d and a an annotation which annotates f and therefore is a subcontext of f, then a also annotates d and is a subcontext of d:

$$a \in sub(f) \text{ and } f \in sub(d) \text{ and } a \text{ annotates } f \Longrightarrow a \in sub(d)$$

Furthermore, to ensure semantic equivalence:

$$(M, w) \models \text{d[f||}p \text{ annoref||]} \Longleftrightarrow (M, w) \models \text{d[}p \text{ annoref f||}p \text{ annoref||]}$$ □

Constraint 10 and the first condition ensures that for an annotation a, R_a is context-serial w.r.t. d, which means that if there is at least one world reachable by d then in this world a is known as an annotation and reaches at least another world. The second condition ensures that d and f have the same knowledge about a and its access probability.

Note that fragment permeability also has consequences for the annotation access probability within fragments; something like

```
d1[ f1|| 0.8 *a1||
    f2|| 0.7 *a1|| ]
```

is not allowed, because in this case, different access probabilities would be transferred to the context d1.

Model of a Basic POLAR Program With the definitions and constraints so far, we are now ready to define the model of a basic POLAR program.

Definition 15 (Model of a basic POLAR program):
An interpretation structure M is a *model* of a basic POLAR program P iff

- $M \models \text{db}[P]$;
- all structured clauses in P (mergedtarget, subcontext, fragment, annoref and reference) are context-valid, and
- constraints 7 to 12 hold.

Note that the context-validity of the structured clauses in P implies that these are valid w.r.t. a model M. We give an exhaustive example of a POLAR program and its corresponding model.

Example 7 (Model of a basic POLAR program): Consider the POLAR program

```
d1[ 0.6 soccer
    0.8 s1[ 0.3 music ]
    0.7 *a1]
a1[ 0.5 football ]
document(d1)
annotation(a1)
```

We obtain the following probabilities and truth value assignments w.r.t. possible worlds contextagent db can reach (let 'd' stand for 'document(d1)', 'a' for 'annotation(d1)', 's' for 'soccer', 'm' for 'music' and 'f' for 'football'; special propositions are not listed here):

w'	$\mu_{db,w_{root}}(w')$	$\pi(w')$(d)	$\pi(w')$(a)	$\pi(w')$(s)	$\pi(w')$(m)	$\pi(w')$(f)
w_1	1.0	*true*	*true*	*unknown*	*unknown*	*unknown*

In w_{root}, the global database agent db reaches exactly one world w_1 in which 'document(d1)', 'a' and 'annotation(d1)' are *true*. For d1, we get:

w'	$\mu_{d1,w_{db}}(w')$	$\pi(w')$(d)	$\pi(w')$(a)	$\pi(w')$(s)	$\pi(w')$(m)	$\pi(w')$(f)
w_{10}	0.024	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_2	0.036	*unknown*	*unknown*	*true*	*unknown*	*unknown*
w_3	0.096	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_4	0.144	*unknown*	*unknown*	*true*	*unknown*	*unknown*
w_5	0.224	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_6	0.336	*unknown*	*unknown*	*true*	*unknown*	*unknown*
w_7	0.056	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_8	0.084	*unknown*	*unknown*	*true*	*unknown*	*unknown*

Access to w_{10} represents the case that 'soccer' is *unknown* (probability is $1 - 0.6 = 0.4$) and we do not access s1 (i.e., s1 does not reach any other world from w_1 and _subpart(s1) is *false*, probability is $1 - 0.8 = 0.2$) and we do not access a1 (a1 cannot reach any other worlds from w_{10}, probability is $1 - 0.7 = 0.3$). So $\mu_{d1,w_1}(w_{10})$ is $0.4 \cdot 0.2 \cdot 0.3 = 0.024$. In w_2, "soccer" is true, but we access neither s1 nor a1, which yields $\mu_{d1,w_{db}}(w_2) = 0.6 \cdot 0.2 \cdot 0.3 = 0.036$. In w_3, 'soccer' is again *unknown*, but this time we access s1, but do not access a1. We obtain $\mu_{d1,w_1}(w_3) = 0.4 \cdot 0.8 \cdot 0.3 = 0.096$. All other cases are calculated in a similar way. What we can read from the table above is the probability that d1 considers 'soccer' to be *true*, which is the sum of the probabilities of all worlds where 'soccer' is assigned the value *true*: $\mu_{d1,w_1}(\{w_2, w_3, w_2, w_2\}) = 0.036 + 0.144 + 0.336 + 0.084 = 0.6$. This is exactly the probability we assigned to 'soccer' in context d1 in our POLAR program. For d1, we obtain the following accessibility relations:

$$\begin{aligned} R_{d1} \quad = \quad & \{(w_{db}, w_{10}), (w_1, w_2), (w_1, w_3), (w_1, w_4), \\ & (w_1, w_5), (w_1, w_6), (w_1, w_7), (w_1, w_8)\} \end{aligned}$$

We continue with contextagent s1, which reaches several worlds from w_3, w_4, w_5 and w_6:

w'	$\mu_{s1,w_3}(w')$	$\pi(w')$(d)	$\pi(w')$(a)	$\pi(w')$(s)	$\pi(w')$(m)	$\pi(w')$(f)
w_{10}	0.7	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_{11}	0.3	*unknown*	*unknown*	*unknown*	*true*	*unknown*

w'	$\mu_{s1,w_4}(w')$	$\pi(w')$(d)	$\pi(w')$(a)	$\pi(w')$(s)	$\pi(w')$(m)	$\pi(w')$(f)
w_{10}	0.7	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_{11}	0.3	*unknown*	*unknown*	*unknown*	*true*	*unknown*

w'	$\mu_{s1,w_5}(w')$	$\pi(w')$(d)	$\pi(w')$(a)	$\pi(w')$(s)	$\pi(w')$(m)	$\pi(w')$(f)
w_{10}	0.7	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_{11}	0.3	*unknown*	*unknown*	*unknown*	*true*	*unknown*

w'	$\mu_{s1,w_6}(w')$	$\pi(w')$(d)	$\pi(w')$(a)	$\pi(w')$(s)	$\pi(w')$(m)	$\pi(w')$(f)
w_{10}	0.7	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_{11}	0.3	*unknown*	*unknown*	*unknown*	*true*	*unknown*

As accessibility relations, we get

$$\begin{aligned} R_{s1} \quad = \quad & \{(w_3, w_{10}), (w_3, w_{11}), (w_4, w_{10}), (w_4, w_{11}), \\ & (w_5, w_{10}), (w_5, w_{11}), (w_6, w_{10}), (w_6, w_{11})\} \end{aligned}$$

Finally, for a1:

w'	$\mu_{a1,w_1}(w')$	$\pi(w')$(d)	$\pi(w')$(a)	$\pi(w')$(s)	$\pi(w')$(m)	$\pi(w')$(f)
w_{10}	0.5	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_9	0.5	*unknown*	*unknown*	*unknown*	*unknown*	*true*

w'	$\mu_{a1,w_5}(w')$	$\pi(w')$(d)	$\pi(w')$(a)	$\pi(w')$(s)	$\pi(w')$(m)	$\pi(w')$(f)
w_{10}	0.5	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_9	0.5	*unknown*	*unknown*	*unknown*	*unknown*	*true*

w'	$\mu_{a1,w_6}(w')$	$\pi(w')$(d)	$\pi(w')$(a)	$\pi(w')$(s)	$\pi(w')$(m)	$\pi(w')$(f)
w_{10}	0.5	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_9	0.5	*unknown*	*unknown*	*unknown*	*unknown*	*true*

w'	$\mu_{a1,w_7}(w')$	$\pi(w')$(d)	$\pi(w')$(a)	$\pi(w')$(s)	$\pi(w')$(m)	$\pi(w')$(f)
w_{10}	0.5	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_9	0.5	*unknown*	*unknown*	*unknown*	*unknown*	*true*

w'	$\mu_{a1,w_8}(w')$	$\pi(w')$(d)	$\pi(w')$(a)	$\pi(w')$(s)	$\pi(w')$(m)	$\pi(w')$(f)
w_{10}	0.5	*unknown*	*unknown*	*unknown*	*unknown*	*unknown*
w_9	0.5	*unknown*	*unknown*	*unknown*	*unknown*	*true*

The accessibility relations of a1 are:

$$\begin{aligned} R_{a1} \quad = \quad & \{(w_5, w_9), (w_5, w_{10}), (w_6, w_9), (w_6, w_{10}), (w_7, w_9), \\ & (w_7, w_{10}), (w_8, w_9), (w_8, w_{10}), (w_1, w_9), (w_1, w_{10})\} \end{aligned}$$

The possible model is depicted in Figure 5.9. □

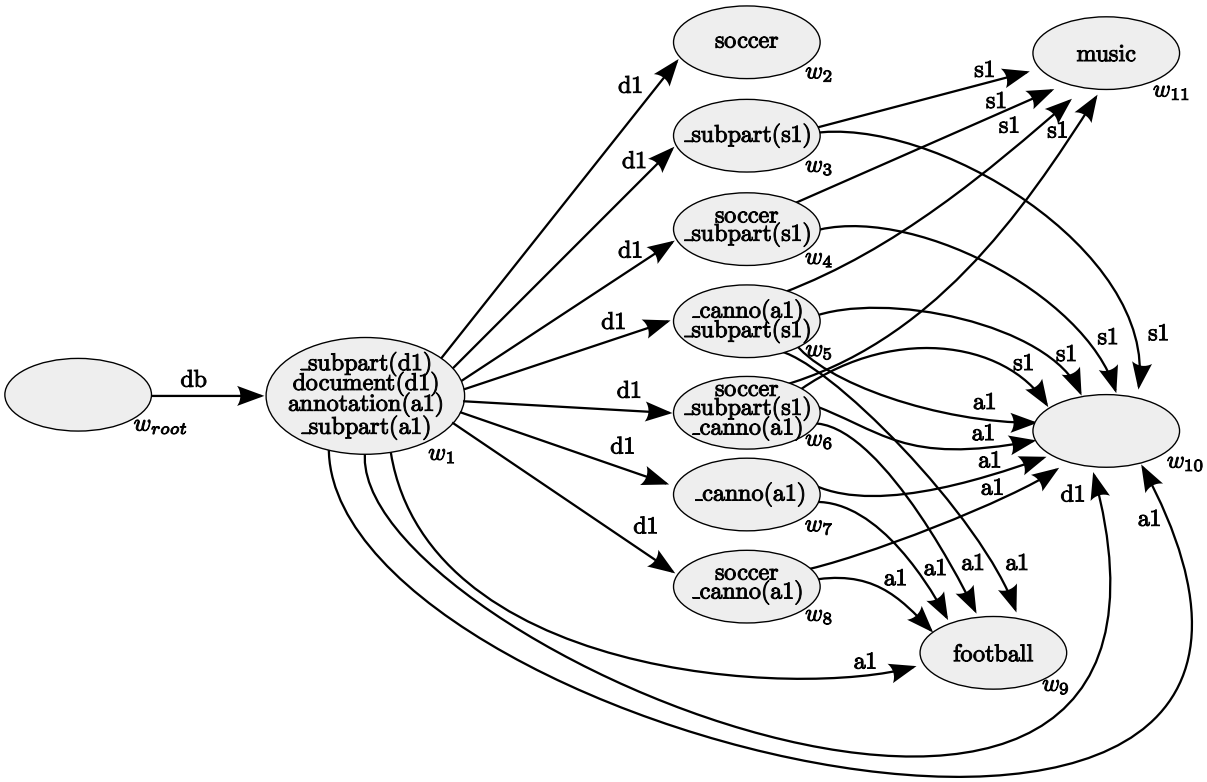


Figure 5.9: A model of the POLAR program in Example 7 (probabilities omitted)

5.2.3 Knowledge Augmentation

With *knowledge augmentation*, we augment the knowledge contained in an object with the one contained in subparts, annotations, referenced objects, merged targets and fragments. The analogy is a reader who reads a document and augments the information she got from the document with the one from annotations, merged targets and referenced objects or, in case of fragments, learns that some parts of a document seem to be important.

The concept of knowledge augmentation was used in POOL and is discussed in Lalmas and Rölleke (2003) and (Rölleke, 1998, chapter 3). Knowledge augmentation in POLAR is based on the one in POOL, but we need to extend some of the POOL concepts presented. We will therefore start with a review of knowledge augmentation in POOL and then discuss the extensions needed by POLAR.

5.2.3.1 Knowledge Augmentation in POOL

Consider the POOL example from Section 4.2.1.2:

```
d1[ 0.9 s1 [ 0.8/0.2 sailing ]
    0.7 s2 [ 0.6/0.4 sailing ]]
```

The document d1 consists of two subparts, s1 and s2, which are accessed with 0.9 and 0.7 probability, respectively. In s1, the term 'sailing' is *true* with 0.8 probability and *false* with 0.2 probability, and it is *true* with a probability of 0.6 and *false* with a probability of 0.4 in s2. The

aim of knowledge augmentation is to calculate the probability that 'sailing' is true or false in the augmented context d1(s1,s2). A possible model of the POOL program above can be seen in Figure 5.10[3].

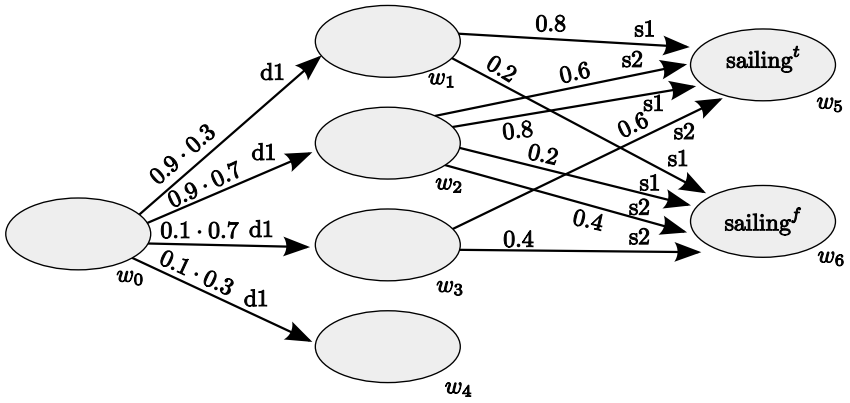


Figure 5.10: A possible model of the POOL program

Basic Idea The idea of knowledge augmentation is as follows: we count every possible case of access and truth value assignments which can happen in the augmented context and for each case, we look up whether the case leads to a *true*, *false*, *inconsistent* or *unknown* truth value assignment for our desired proposition. Each such case can occur with a certain probability. The probability of a proposition to be true is then the sum over all cases where the proposition becomes true in the augmented context. Take, for example, the case that we access s1 from d1 (prob. is 0.9) and 'sailing' is *true* (prob. is 0.8) and we also access s2 (prob. 0.7), but now 'sailing' is *false* (prob. 0.4). The probability of this case is $0.9 \cdot 0.8 \cdot 0.7 \cdot 0.4 = 0.1$. Since in this case we have contradicting evidence ('sailing' is *true* in s1, but *false* in s2), we assign 'sailing' the truth value *inconsistent* in the augmented context for this case. Note that for determining the truth value assignment and the probability of this case, we used a subtree of the structure in Fig. 5.10, namely the one containing the edge between w_0 and w_2, the edge between w_2 and w_5 labelled "s1" and the one between w_2 and w_6 labelled "s2". In general, each subtree in the model represents a possible case. The idea is to define a function which detects all possible subtrees (and therefore detects the possible cases) for an augmented context, and then assigns a probability to each of these cases and truth values for each proposition in the subtree. Each subtree, i.e. each case, is regarded as a world in its own right, which is accessed with the assigned probability and contains the derived truth values.

G-Reachability In the following, we regard every possible case as an own world w^* which d1(s1,s2) accesses with the probability of the corresponding case (we use the '*' superscript to refer to such worlds). Each such case is a subtree in our interpretation structure. This subtree is built upon the notion of *G-reachability* (Fagin et al., 1995, p. 23) – a world w' is *G-reachable* from a world w if there is a path in the graph from w to w' whose edges are labelled by members of the group G. For instance, the world w_5 in Fig. 5.10 is G-reachable from w_0 for $G = \{\text{d1}, \text{s1}\}$ or $G = \{\text{d1}, \text{s2}\}$, but not for $G = \{\text{s1}, \text{s2}\}$

[3]Since POOL only knows subparts as subcontexts, no special propositions or other means to specify the relation type are needed.

Now each subtree which establishes a case is called a *G-agent-world-tree*. The nodes of such a tree are a tuple (d, w), where d is the contextagent that accesses w. Formally, a G-agent-world-tree is a tuple (w, S) where w is the head of the tree and S the set of subtrees. Figure 5.11 shows three example trees of worlds representing cases (which are described below). For example, the tree underlying w_1^* and the group $G = (\mathrm{d1}, \mathrm{s1}, \mathrm{s2})$ is $((\mathrm{d1}, w_2), \{((\mathrm{s1}, w_5), \{\}), ((\mathrm{s2}, w_5), \{\})\})$. We can formally describe a function which creates G-agent-world-trees as the set of trees for a

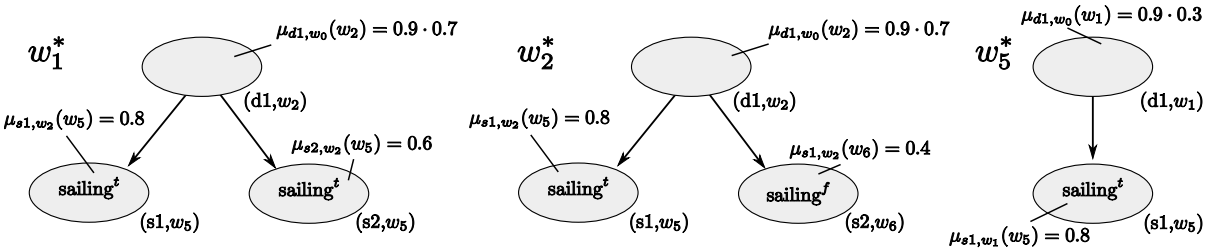


Figure 5.11: Some selected G-agent-world-trees

context and a group (see also (Rölleke, 1998, p. 44)):

$$trees((c, w_0), dG) := \{((d, w), S) | w \in R_d(w_0) \ \wedge \ ((d, w), S) \in trees((d, w), G)\} \tag{5.1a}$$

$$\begin{aligned} trees((d, w), G_{n+1}) := \{((d, w), S) \,|\, \exists S_n, t : ((d, w), S_n) \in trees((d, w), G_n) \\ \wedge\, t \in trees((d, w), s_{n+1}) \wedge S = S_n \cup \{t\}\} \end{aligned} \tag{5.1b}$$

G is a group, G_{n+1} a group $(s_1, \ldots, s_{n+1})$ of subcontexts and $G_n = (s_1, \ldots, s_n)$. For our example above, we need to calculate $trees((\mathrm{db}, w_0), \mathrm{d1(s1,s2)})$ to get our G-agent-world-trees.

The *trees* function calculates all trees, that is cases or worlds w^*. For each tree, we need its probability, and the augmented truth value assignments for the propositions in the respective world. For the former, we apply the notion of augmented and united probabilities. For the latter, we extend the present truth value assignment function.

Augmented and United Probabilities For each tree, we apply the definition for *augmented* and for *united truth probabilities* (μ_A and μ_U, respectively)

$$\begin{aligned} \mu_{A,w_0}(((d, w), S)) &:= \mu_{d,w_0}(w) \cdot \mu_U(((d, w), S)) \\ \mu_U(((d, w), S) &:= \prod_{t \in S} \mu_{A,w}(t) \end{aligned}$$

to calculate the probability of each tree. For w_1^*, for instance, we compute

$$\begin{aligned} &\mu_{A,w_0}(((\mathrm{d1}, w_2), \{((\mathrm{s1}, w_5), \{\}), ((\mathrm{s2}, w_6), \{\})\})) = \\ &\quad \mu_{d1,w_0}(w_2) \cdot \mu_U(((\mathrm{d1}, w_2), \{((\mathrm{s1}, w_5), \{\}), ((\mathrm{s2}, w_5), \{\})\})) = \\ &\quad \mu_{d1,w_0}(w_2) \cdot \mu_{A,w_2}(((\mathrm{s1}, w_5), \{\})) \cdot \mu_{A,w_2}(((\mathrm{s2}, w_5), \{\})) = \\ &\quad \mu_{d1,w_0}(w_2) \cdot \mu_{s1,w_2}(w_5) \cdot \mu_{s2,w_2}(w_5) = \\ &\quad 0.9 \cdot 0.7 \cdot 0.8 \cdot 0.6 = 0.3024 \end{aligned}$$

This is the probability that w_1^* is accessed from w_0, or in other words, that the augmented contextagent d1(s1,s2) considers w_1^* possible. This probability is therefore denoted $\mu_{d1(s1,s2),w_0}(w_1^*)$.

Truth Value Assignment Finally, to get the truth value for a proposition in a G-agent-world-tree like w_1^*, the truth value assignment function $\pi(w^*)(\varphi)$ calculates the set union over the truth values of all worlds in the G-agent-world-tree. For $\pi(w_1^*)(\text{sailing})$ we thus get the value *true*. When talking of a "set union over truth values", this adheres to the view of the four possible truth values as sets (see page 57). For a G-agent-world-tree w^* containing the worlds $(w_1, \ldots, w_n)$, it is

$$\pi(w^*)(\varphi) = \bigcup_{w \in (w_1,\ldots,w_n)} \pi(w)(\varphi)$$

For the worlds depicted in Figure 5.11, we obtain

$$\begin{aligned}
\pi(w_1^*)(\text{sailing}) &= \pi(w_2)(\text{sailing}) \cup \pi(w_5)(\text{sailing}) \cup \pi(w_5)(\text{sailing}) \\
&= \mathit{unknown} \cup \mathit{true} \cup \mathit{true} = \mathit{true} \\
\pi(w_2^*)(\text{sailing}) &= \pi(w_2)(\text{sailing}) \cup \pi(w_5)(\text{sailing}) \cup \pi(w_6)(\text{sailing}) \\
&= \mathit{unknown} \cup \mathit{true} \cup \mathit{false} = \mathit{inconsistent} \\
\pi(w_5^*)(\text{sailing}) &= \pi(w_1)(\text{sailing}) \cup \pi(w_5)(\text{sailing}) \\
&= \mathit{unknown} \cup \mathit{true} = \mathit{true}
\end{aligned}$$

For our example case w_2^*, we found evidence for *true* in w_5, *false* in w_6 and *unknown* in w_2, which leads to $\{t\} \cup \{f\} \cup \emptyset = \{t, f\} = \mathit{inconsistent}$.

Example 8 (Knowledge augmentation in POOL): For the example POOL program above we obtain the following worlds, probabilities and truth value assignments:

w^*	$\mu_{d1(s1,s2),w_0}(w^*)$	$\pi(w^*)(\text{sailing})$
w_1^*	$0.9 \cdot 0.7 \cdot 0.8 \cdot 0.6$	$\mathit{true} = \mathit{true} \cup \mathit{true}$
w_2^*	$0.9 \cdot 0.7 \cdot 0.8 \cdot 0.4$	$\mathit{inconsistent} = \mathit{true} \cup \mathit{false}$
w_3^*	$0.9 \cdot 0.7 \cdot 0.2 \cdot 0.6$	$\mathit{inconsistent} = \mathit{false} \cup \mathit{true}$
w_4^*	$0.9 \cdot 0.7 \cdot 0.2 \cdot 0.4$	$\mathit{false} = \mathit{false} \cup \mathit{false}$
w_5^*	$0.9 \cdot 0.3 \cdot 0.8$	*true*
w_6^*	$0.9 \cdot 0.3 \cdot 0.2$	*false*
w_7^*	$0.1 \cdot 0.7 \cdot 0.6$	*true*
w_8^*	$0.1 \cdot 0.7 \cdot 0.4$	*false*
w_9^*	$0.1 \cdot 0.3$	*unknown*

w_1^* denotes the case that we access s1 and 'sailing' is *true* in s1 and we access s2 and 'sailing' is *true* in s2. w_2^* represents the case that we access s1 and 'sailing' is *true* in s1 and we access s2 and 'sailing' is *false* in s2. The world w_3^* stands for the case that that we access s1 and 'sailing' is *false* in s1 and we access s2 and 'sailing' is *true* in s2. In world w_4^*, we access both s1 and s2, but 'sailing' is *false* in both of them. w_5^* denotes the case that we access s1 but not s2 and 'sailing' is *true* in s1. In w_6^*, we access s1 but not s2 and 'sailing' is *false* in s1. w_7^* and w_8^* represent the cases that we do not access s1 but access s2; in w_7^* 'sailing' is *true* in s2, whereas in w_8^* 'sailing' is *false* in s2. Finally, w_9^* denotes the case that we neither access s1 nor s2.

Note that all cases are disjoint. $w_1^*, \ldots, w_9^*$ are the worlds reached by the (augmented) contextagent d1(s1,s2) from w_0, as Fig. 5.12 illustrates. We can therefore define an accessibility relation for augmented contextagents (here: $R_{d1(s1,s2)}$) similar to those for non-augmented ones. Furthermore, above probabilities sum to 1, so $\mu_{d1(s1,s2),w_0}$ is a probability mass function (see (Rölleke, 1998, p. 61) for a proof). Then the probabilities for *true*, *false*, *inconsistent* and

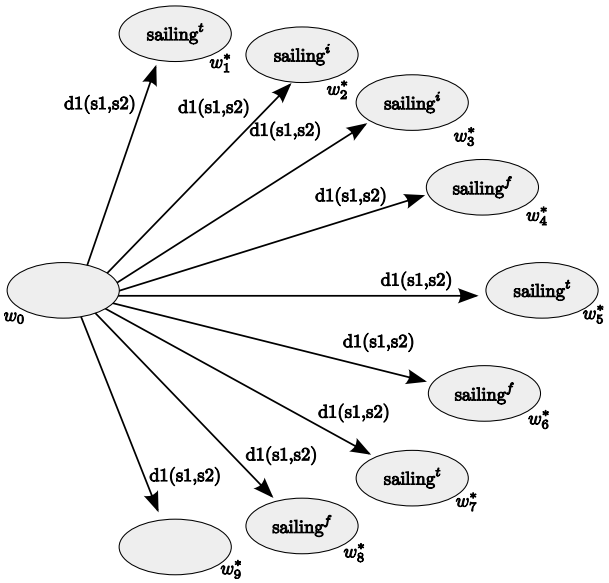


Figure 5.12: Worlds reachable by the augmented contextagent d1(s1,s2) (probabilities omitted)

unknown in the augmented context d1(s1,s2) are the sum of the probabilities of the cases producing the corresponding truth value:

$$
\begin{aligned}
&(M, w_0) \models \text{d1(s1,s2)[0.5604/0.1324/0.2772/0.03 sailing]} \Longleftrightarrow \\
&\quad 0.5604 = \mu_{d1(s1,s2),w_0}(\{w^* | w^* \in R_{d1(s1,s2)}(w_0) \wedge (M, w^*) \models_t \text{sailing}\}) \quad \text{and} \\
&\quad 0.1324 = \mu_{d1(s1,s2),w_0}(\{w^* | w^* \in R_{d1(s1,s2)}(w_0) \wedge (M, w') \models_f \text{sailing}\}) \quad \text{and} \\
&\quad 0.2772 = \mu_{d1(s1,s2),w_0}(\{w^* | w^* \in R_{d1(s1,s2)}(w_0) \wedge (M, w^*) \models_i \text{sailing}\}) \quad \text{and} \\
&\quad 0.03 = \mu_{d1(s1,s2),w_0}(\{w^* | w^* \in R_{d1(s1,s2)}(w_0) \wedge (M, w^*) \models_u \text{sailing}\})
\end{aligned}
$$

□

We presented the knowledge augmentation approach in POOL. Although a POOL program like the one above is a POLAR program as well, we need to extend the knowledge augmentation approach in order to cope with the different kinds of subcontexts we find in POLAR.

5.2.3.2 Knowledge Augmentation in POLAR

In Chapter 2 we discussed some facets of annotations. One of the main findings was that annotations can be content about content, not extending the content in the annotated document, but saying something about this content. This motivated the introduction of meta level annotations. On the other hand, annotations can be extensions of the content in the annotated document, which led to the introduction of content level annotations. In our considerations in Chapter 3, we also discussed fragments and annotation targets, which led to the definition of fragments and merged annotation targets as special kind of subcontexts. Fragments are important items to satisfy certain information needs regarding annotations, as shown in the last chapter. One of the main aims of the POLAR framework is to search for documents and annotations. To determine the relevance of an object (document or annotation) w.r.t. a query, we also want to use the augmented context of this object, instead of only considering the object itself. This context is made of the subparts, the merged annotation target, the fragments,

the referenced objects, and of course the annotations of the object. In contrast to POOL, where we only dealt with subparts, we need to be careful when considering the various types of subcontext we find in an annotation scenario. Can we just augment the content of an object with the one we find in every possible (augmented) subcontext, like POOL does? Each kind of subcontext has some specific features which influence the propagation of its knowledge to the supercontext. We therefore continue with a discussion of the various kinds of subcontexts and their role w.r.t. knowledge augmentation.

Merged Annotation Targets Merged annotation targets provide the context for the annotation content. Consider a fragment saying "we went to Paris" and an elaboration annotating this fragment saying "The one in Texas, not in France". We need the knowledge that "the one" means "Paris" to determine that the annotation is about "Paris". So we need to augment our annotation with the content of the merged annotation targets, as otherwise the annotation might not be understandable without considering the annotated object. On the other hand, merged targets of subcontexts should not be augmented, as a small example illustrates:

```
d1[ 0.6 t1  f1|| 0.6 t1  *a1 || ]
a1[ m1< 0.6 t1 >
    0.7 t2  f2|| 0.7 t2  *a3 ||]
a2[ 0.7 t3  *a3]
a3[ m3< 0.7 t2  0.7 t3 >]
```

When calculating the augmented context a1(m1,f2,a3), we should consider m1, because this introduces t1 to a1. We should also consider a3, since it annotates a1 through f2. Should we propagate the knowledge in the merged annotation target m3 as well? m3 consists of propositions found in a2 and a1. The propositions in a2 do not have anything to do with the ones in a1 in the first place[4]. The propagation of t2 in m3 to a1(m1,f2,a3) would not introduce new information to a1(m1,f2,a3) – the fact that t2 seems to be important is already covered in f2. Finally, by design the main purpose of m3 is to provide a context for understanding the propositions in a3 – it was never meant to extract contextual information from m3 for other objects than a3. These are the reasons why we should compute a1(m1,f2,a3) and not a1(m1,f2,a3(m3)). The same argument holds for the augmented context of d1, which should be d1(f1,a1(a3)) and not d1(f1,a1(m1,a3)) or even d1(f1,a1(m1,a3(m3))). So we need to prohibit that merged annotation targets are propagated for an augmented context cG except for the case that they are the merged annotation target of c.

Content Annotations If we are searching for documents about "Paris in Texas", how can we determine in the example above that the fragment or the document the fragment belongs to is relevant to this query, as only "Paris" might be mentioned in this document? It seems to be beneficial if we augment the knowledge in our document with the one in its annotations, since it propagates the knowledge in annotations to the supercontext, where it is aggregated.

Fragments Fragments might also play an interesting part. In case of highlighting, such fragments mark important parts of a document. But even if fragments are no explicit highlighting,

[4]It can be argued that a1 and a2 are in a way related since they are both annotated by a3. But this kind of relation needs further discussion and evaluation and will be neglected for now, so we completely ignore a2 in the augmented context of a1.

they are the parts of the document others found important enough to annotate. This information should be considered as well, so this is the reason why we augment our knowledge about the document with the knowledge in its fragments. Note that propagating the content of fragments does not introduce new propositions to the supercontext, but can raise our belief that the proposition is true (or false, respectively) in the supercontext by adapting the four-valued probability of the proposition.

Polarity Another issue of content annotations is their polarity. What does it mean for the augmentation if we have an annotation with negative polarity? In our interpretation of negative polarity in a retrieval scenario, we said that such an annotation gives us evidence that the annotated objects should not be indexed with the terms appearing in a negative annotation. So if a proposition is true in a negative annotation, it gives us evidence that it is false in the augmented context, and vice versa.

Meta annotations If we say that we want to augment the knowledge of an object with the knowledge in its annotations, then there is one exception, which are meta level annotations. They should not be considered for knowledge augmentation. The reason is that meta level annotations make assertions about the content of an object, but they do not expand its content. For example, a meta level annotation saying "Good comment!" does not mean that the annotated object is about "comment". So our conclusion is that we exclude meta annotations from knowledge augmentation.

Subparts Finally, we consider the subparts of an object for augmentation to allow for structured document retrieval like in POOL.

The discussion above makes clear that we cannot just declare everything as a subcontext and reuse the knowledge augmentation routines from POOL. We have to take several special cases into account, which are:

1. Subcontexts of fragments and meta annotations must be ignored;
2. Merged annotation targets must not be propagated if they belong to a subcontext of the context to augment;
3. We need to include the effect of negative polarity.

For integrating polarity, we need to code the information about the polarity of an annotation in the knowledge augmentation process into the G-agent-world-tree. We continue with the discussion of how to handle polarity before describing how to handle merged annotation targets, fragments and meta annotations correctly.

Handling Polarity To cope with polarity, the idea is to extend G-agent-world-trees to *G-agent-world-polarity-trees*. Nodes in these trees not only convey information about which contextagent entered the corresponding world, but also about the polarity the world has. If the contextagent represents a negative content annotation in the world it entered the current world from, the polarity of the entered world is the negation of the polarity of the source world. So we have to define a polarity function *pol* which calculates the new polarity of a world-agent-tuple:

Definition 16 (Polarity function *pol*):
Let $\rho \in \{0, 1\}$ be the *polarity* of a world-agent tuple (W, a). 0 stands for negative, 1 for positive polarity. The function *pol* reverses the polarity in case a is a negative content annotation and does nothing in any other case:

$$pol(w, a, \rho) = \begin{cases} 1 & \text{if } (M, w) \models_t \text{\_negcanno}(a) \text{ and } \rho = 0, \\ 0 & \text{if } (M, w) \models_t \text{\_negcanno}(a) \text{ and } \rho = 1, \\ \rho & \text{else.} \end{cases}$$

Based on the polarity function and equation (5.1) describing *trees* to create G-agent-world-trees, we create G-agent-world-polarity-trees in POLAR using a new function *trees*.

Definition 17 (POLAR *trees* function):

$$\begin{aligned} trees((d, w_0, \rho), aG) := \{((a, w, \rho'), S) \mid & w \in R_d(w_0) \wedge \\ & \rho' = pol(w_0, a, \rho) \wedge \\ & ((a, w, \rho'), S) \in trees((a, w, \rho'), G)\} \end{aligned} \tag{5.2a}$$

$$\begin{aligned} trees((d, w, \rho), G_{n+1}) := \{((d, w, \rho), S) \mid & \exists S_n, t : ((d, w, \rho), S_n) \in trees((d, w, \rho), G_n) \\ & \wedge t \in trees((d, w, \rho), s_{n+1}) \wedge S = S_n \cup \{t\}\} \end{aligned} \tag{5.2b}$$

Furthermore, $trees((d, w, \rho), ()) = ((d, w, \rho), \{\})$ for empty groups. (5.2a) toggles the polarity flag each time it encounters a negative content annotation. In particular, if a negative content annotation is a subcontext of another negative content annotation, the flag is changed back to positive polarity.

We give an example to explain POLAR's *trees* function.

Example 9 (*G*-agent-world-polarity-trees): Consider the following POLAR program:

```
d1[0.8 political -*a1]
a1[0.7 political]
```

A possible model for the POLAR program can be seen in Fig. 5.13. *trees*((db,w_0,1), d1(a1))

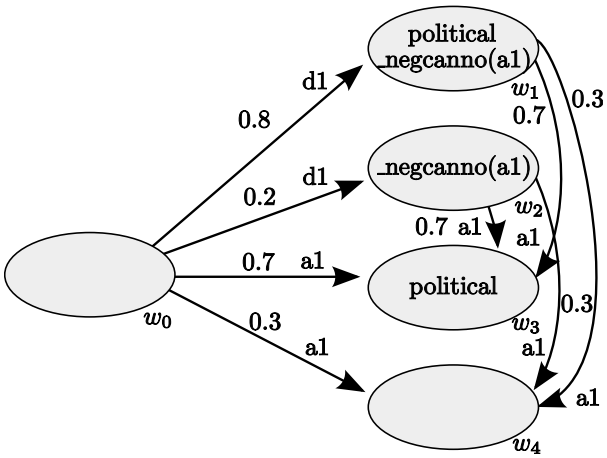


Figure 5.13: A model for the negative polarity example

calculates the following subtrees:

$$
\begin{aligned}
trees((\mathrm{a1},w_4,0),()) &= \{((\mathrm{a1},w_3,0),\{\}),((\mathrm{a1},w_4,0),\{\})\}\\
trees((\mathrm{a1},w_3,0),()) &= \{((\mathrm{a1},w_3,0),\{\}),((\mathrm{a1},w_4,0),\{\})\}\\
trees((\mathrm{d1},w_1,1),\mathrm{a1}) &= \{((\mathrm{a1},w_3,0),\{\}),((\mathrm{a1},w_4,0),\{\})\}\\
trees((\mathrm{d1},w_2,1),\mathrm{a1}) &= \{((\mathrm{a1},w_3,0),\{\}),((\mathrm{a1},w_4,0),\{\})\}\\
trees((\mathrm{d1},w_1,1),(\mathrm{a1})) &= \{((\mathrm{d1},w_1,1),\{((\mathrm{a1},w_3,0),\{\})\}),\\
&\quad\ ((\mathrm{d1},w_1,1),\{((\mathrm{a1},w_4,0)),\{\})\})\}\\
trees((\mathrm{d1},w_2,1),(\mathrm{a1})) &= \{((\mathrm{d1},w_2,1),\{((\mathrm{a1},w_3,0),\{\})\}),\\
&\quad\ ((\mathrm{d1},w_2,1),\{((\mathrm{a1},w_4,0)),\{\})\})\}\\
trees((\mathrm{db},w_0,1),\mathrm{d1}(\mathrm{a1})) &= \{((\mathrm{d1},w_1,1),\{((\mathrm{a1},w_3,0),\{\})\}),\\
&\quad\ ((\mathrm{d1},w_1,1),\{((\mathrm{a1},w_4,0)),\{\})\})\\
&\quad\ ((\mathrm{d1},w_2,1),\{((\mathrm{a1},w_3,0),\{\})\}),\\
&\quad\ ((\mathrm{d1},w_2,1),\{((\mathrm{a1},w_4,0)),\{\})\})\}
\end{aligned}
$$

While processing, $trees((\mathrm{db},w_0,1),\mathrm{d1}(\mathrm{a1}))$ invokes $trees((\mathrm{d1},w_2,1),(\mathrm{a1}))$ and $trees((\mathrm{d1},w_1,1),(\mathrm{a1}))$. $trees((\mathrm{d1},w_2,1),(\mathrm{a1}))$ invokes $trees((\mathrm{d1},w_2,1),\mathrm{a1})$ and $trees((\mathrm{d1},w_1,1),(\mathrm{a1}))$ calls $trees((\mathrm{d1},w_1,1),\mathrm{a1})$. Finally, $trees((\mathrm{d1},w_2,1),\mathrm{a1})$ and $trees((\mathrm{d1},w_1,1),\mathrm{a1})$ both invoke $trees((\mathrm{a1},w_3,0),())$ and $trees((\mathrm{a1},w_4,0),())$.

The resulting trees can be seen in Figure 5.14. Each node of a tree contains information about the corresponding polarity. World w_1^* represents the case that 'political' is true in d1 and a1, world w_2^* models the case that 'political' is true only in d1, world w_3^* stands for 'political' being true only in a1 and in w_4^*, 'political' is neither true in d1 nor in a1. In all cases, d1 accesses a1. □

To get the final truth value for every proposition in any of the worlds $w_1^*,\ldots,w_4^*$, we again use the union of the truth values of the proposition in each single world $w_0,\ldots,w_4$. To capture the polarity, we need to extend the truth value function π to consider polarity as well. Before

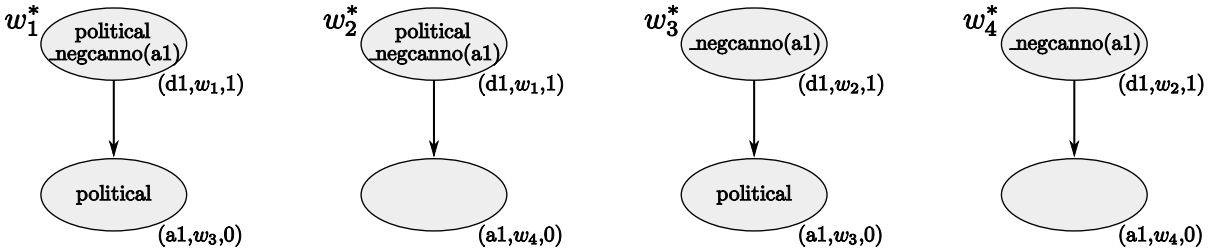


Figure 5.14: Output of $trees((\text{db},w_0,1), \text{d1}(\text{a1}))$

we do that, we present the negation of a four-valued truth value. We follow the definition in (Rölleke, 1998, p. 71) (see also Belnap (1977)), which is based on the set interpretation of truth values introduced on page 57.

Definition 18 (Negation of four-valued truth values):
For a proposition φ and an arbitrary world w, t is an element of the truth value assignment $\pi(w)(\text{not}\,\varphi)$ iff f is an element of $\pi(w)(\varphi)$. f is an element of the truth value assignment $\pi(w)(\text{not}\,\varphi)$ iff t is an element of $\pi(w)(\varphi)$. For example, if $\pi(w)(\varphi) = \mathit{true} = \{t\}$, then $\pi(w)(\text{not}\,\varphi) = \{f\} = \mathit{false}$. We gain the following truth value definitions:

$\pi(w)(\varphi)$	$\pi(w)(\text{not}\,\varphi)$
true	*false*
false	*true*
inconsistent	*inconsistent*
unknown	*unknown*

Based on the definition of the negation, we define a truth value assignment function which takes the polarity of a world into account.

Definition 19 (Polarity-based truth value assignment):
The polarity-based truth value assignment function assigns the original truth value of a proposition φ in a world w if w has positive polarity in the G-agent-world-polarity-tree. It assigns the negated value in case of negative polarity.

$$\pi_p(w,\rho)(\varphi) = \begin{cases} \pi(w)(\varphi) & \text{if } \rho = 1, \\ \pi(w)(\text{not}\,\varphi) & \text{if } \rho = 0. \end{cases}$$

Let w^* be a world represented by a G-agent-world-polarity-tree and let $(w_1,\ldots,w_n)$ be the worlds in this tree. The truth value assignment function w^* takes the union of the polarity-based truth value assignment functions for $(w_1,\ldots,w_n)$ to derive a final truth value for a proposition φ in w^*:

$$\pi(w^*)(\varphi) = \bigcup_{w\in(w_1,\ldots,w_n)} \pi_p(w,\rho)(\varphi)$$

Taking Example 9, we obtain:

w^*	$\mu_{d1(a1),w_0}(w^*)$	$\pi(w^*)$(political)
w_1^*	$0.8 \cdot 0.7$	$inconsistent = true \cup false$
w_2^*	$0.8 \cdot 0.3$	$true$
w_3^*	$0.2 \cdot 0.7$	$false$
w_4^*	$0.2 \cdot 0.3$	$unknown$

The negative polarity of a1 in d1 directly influences the truth value assignments in the worlds w_1^* and w_3^* – these were both *true* if a1 was a content annotation instead of a negative content annotation. Finally, we get

```
d1(a1)[0.24/0.14/0.56/0.06 political]
```

for the augmented context d1(a1). The fact that a1 is a negative content annotation lowers our confidence that 'political' is true, and it raises the probability that it is false. But mostly, it raises the probability that we get inconsistent information regarding this term.

We need to handle the peculiarities of the different subcontexts, as discussed above. Additionally, we need to discuss access from multiple contexts as well as possible cycles in our interpretation structure.

Access from multiple contexts Consider the following POLAR program:

```
d1 [ s1 [ 0.5 *a1 ]
     s2 [ 0.4 *a1 ]]
a1 [...]
```

a1 annotates both subparts s1 and s2 of d1. If we augment d1, should we access a1 twice, or should we even ignore one access of a1? And if, which one should be ignored, the access from s1 or the one from s2? In the above example, we can identify the cases that a1 is accessed from both s1 and s2 (with probability of $0.5 \cdot 0.4 = 0.2$), from s1 only ($0.5 \cdot 0.6 = 0.3$ probability), from s2 only (probability of $0.5 \cdot 0.4 = 0.2$) and the last case is that a1 is not accessed at all (with probability of $0.5 \cdot 0.6 = 0.3$). All these cases occur with a certain probability. We may argue that at least the first case should somehow be forbidden, so that a1 should be accessed only once during knowledge augmentation. There is an argument against it: due to the fact that a1 annotates several subparts of s1, a1 seems to be important for all of them. So we allow the case that a1 is accessed more than once, also for the sake of keeping our model pure. If we just neglect the first case, the probability of all considered cases does not sum up to 1, but to $0.3 + 0.2 + 0.3 = 0.8$. So $\pi_{d1(s1(a2),s2(a2)),w}$ would not be a probability mass function, which in a way spoils our model so far. Since annotation-based information retrieval is the main application of our approach, future evaluations should show the effect of our decision on retrieval quality.

Avoiding Cycles In Section 3.2 we showed that the structured annotation hypertext is acyclic. Unfortunately, this does not hold with the access structures in POLAR. While the access relation has the same direction as the references link in the structured annotation hypertext (from the annotation to the referenced object), the hasAnnotationTarget link has the opposite direction than the access relation – hasAnnotationTarget goes from the annotation to the annotated object, whereas annotation access (access to subcontexts which are content annotations) goes

from the annotated object to the annotation. Therefore, we may get undesirable cycles in our interpretation structure, as the following example shows.

```
d1[ *a1 ]
a1[ 0.8 *a2 ]
a2[ =>d1 ]
```

d1 is annotated by a1, which in turn is annotated by a2, where a2 is accessed with 0.8 probability. a2 references d1 again. If we ask for the full augmented context for d1, we would get something like d1(a1(a2(d1(a1(a2(...)))))) which repeats itself all over again. In terms of our interpretation structure, the situation is illustrated in Fig. 5.15. Contextagent d1 accesses the

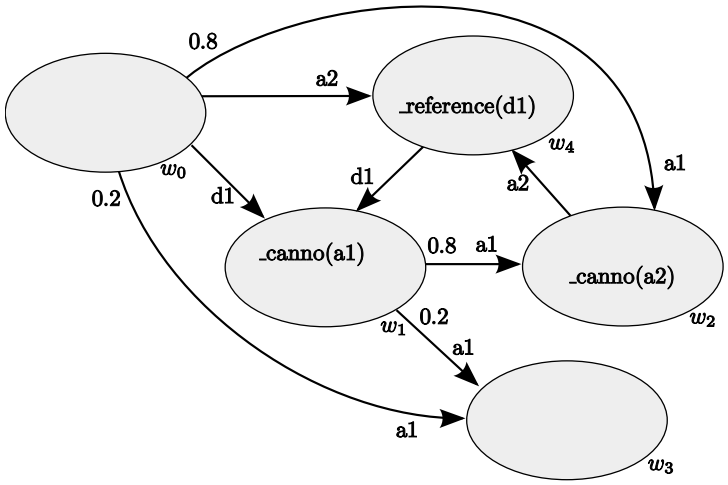


Figure 5.15: A cycle in the interpretation structure

world w_1 from which contextagent a1 either accesses w_2 or w_3. In the latter world, no agent accesses another world any more. In the former world, a2 can reach w_4, from where d1 again accesses w_1. When calculating, e.g., the augmented context of a1, *trees* would visit w_2, then w_4 and then w_1. After that, *trees* would visit w_2 and w_3. We do not want both cases; if w_2 is revisited, we would propagate the information in w_2 twice and even continue the cycle. But we also do not want to enter w_3, because w_3 stands for the case that a2 is not accessed from a1. But having visited w_2 before, we already reflected the case that a2 is reached from a1, so by entering w_3, we would consider a case which is disjoint to the one we already considered, and this is impossible. So we must avoid that a contextagent is active more than once in a path when performing knowledge augmentation with the *trees* function. We do so by defining valid augmented context expressions.

Valid Augmented Context Expressions Augmented context expressions control the behaviour of the *trees* function. The idea is now to create augmented context expressions which reflect the considerations so far. We need an algorithm which builds a valid augmented context expression for a contextagent and a world. For the example illustrated in Fig. 5.15, the algorithms should yield the augmented context expression d1(a1(a2)) for d1 in w_0. This expression is the used to calculate the augmented context of d1 using *trees* and the corresponding truth value assignment functions.

Parameters: Interpretation structure M, context to augment c, world w, path p
Return: Augmented context expression $expr_c$

1: $level := |p|$
2: $expr_c := c$
3: **if** $w \not\models$ _fragment(c) **then**
4: $SUB := \{s | \exists w' \in R_c(w) \wedge$
$(w' \models$ _subpart(s) $\vee\ w' \models$ _canno(s) $\vee\ w' \models$ _negcanno(s) $\vee$
$w' \models$ _reference(s) $\vee w' \models$ _fragment(s) $\vee (w' \models$ _mtarget(s) $\wedge level = 0))\}$
5: **if** $SUB \neq \emptyset$ **then**
6: $p :=$ addToPath(p,c)
7: **for all** $s \in SUB$ **do**
8: **if** !contained(s,p) **then**
9: Select w' with $w' \in R_c(w) \wedge R_s(w') \neq \emptyset$
10: $expr_c :=$ concat($expr_c$,createAugmContextExpr(M,s,w',p))
11: **end if**
12: **end for**
13: **end if**
14: **end if**

Algorithm 1: Algorithm **createAugmContextExpr** to create valid augmented context expressions

Algorithm 1 shows the algorithm **createAugmContextExpr** to create valid augmented context expressions given an interpretation structure M, a context to augment c and a world w. Initially, the algorithm is invoked with an empty path p. Beginning from w, the algorithm performs a depth first traversal through the connected worlds and according to the conditions discussed above. Line 1 extracts the current level; $|p|$ is the length of the path p and 0 if the path is empty. The second line initialises the expression to be returned with the context c. The third line checks if c is known to be a fragment in w. Since we do not want to augment fragments, we only continue if this is not the case. Line 4 creates the set SUB of all considerable direct subcontexts of c. These are all subcontexts which are known to be a subpart, (negative) content annotation, reference, fragment or merged target in a world accessible by c from w. Merged targets are only considered if the level is 0, which means only if they are a direct subcontext of the context to augment. If this set is not empty (line 5), we first add the current context to the path (l. 6). We then loop over all subcontexts (l. 7) and continue with the next subcontext s if it is not contained in the path yet (line 8). This way, we prohibit that a context is its own subcontext, which would indicate a cycle. Line 9 selects an arbitrary world w' which is reachable from w by c and from which the subcontext s reaches other worlds. In line 10, the algorithm is again invoked with s as the context to augment, the starting world w' and the current path p (note that p is not empty now; this means that a merged annotation target of s would not be considered any more). The resulting augmented context expression is concatenated with the current one.

Going back to the example in Fig. 5.15, createAugmContextExpr(M,d1,w_0,()) returns the desired augmented context expression d1(a1(a2)). In a first recursion, createAugmContextExpr(M,a1,w_1,(d1)) is invoked. This calls createAugmContextExpr(M,a2,w_2,(d1,a1)). The processing stops here since !con-

tained(d1,(d1,a1)) in line 8 fails. The algorithm always terminates since it either detects a cycle or there are no suitable subcontexts any more.

In this subsection we introduced a model for basic POLAR programs based on Kripke structures. We also discussed knowledge augmentation and the special cases to consider in POLAR compared to POOL. To finish our discussion about the POLAR semantics, we will now present the semantics of queries and rules.

5.2.4 Queries and Rules

We present the semantics of queries and rules as specified in Figure 5.2. We commence with the discussion of queries, which includes the definition of some new syntactic constructs which were not part of the basic POLAR knowledge modelling, but can be applied in queries and also rules. After having introduced queries, we are going to discuss rules.

5.2.4.1 Queries

As shown in Figure 5.2 on page 92, a query consists of a subgoal list. Each subgoal can be a fact goal consisting of an atom or its negation. Subgoals can also be complex queries containing both facts and structure. Complex queries support several information needs arising in an annotation scenario. When processing queries, the inference engine returns the instantiations of the variables which return a probability > 0 for a subgoal list. We begin our discussion with simple queries consisting of fact goals and continue with context subgoals.

Database Queries Database queries are realised by fact goals. Fact goals are atoms or negated atoms. An atom can be a term, a classification or an attribute, where instead of an object ID, there may be a variable. Consider the simple POLAR program

```
schalke
footballclub(schalke)
0.7/0.3 peter.supports(schalke)
```

We have the term 'schalke' and learn that Schalke is a football club. We guess that Peter supports Schalke, but we are not sure, therefore the 0.7 probability that he supports Schalke and the 0.3 probability that he does not. A POLAR query

```
?- schalke
```

returns

```
1.0 ()
```

which is the probability that 'schalke' is true in the db-context. We might ask for the clubs Peter supports (variables are preceded by a capital letter in POLAR):

```
?- peter.supports(C)
```

and gain

```
0.7 (schalke)
```

The system substitutes `C` with 'schalke' and 0.7 is the probability that 'schalke' is *true*. The query for the clubs Peter does not support

```
?- !peter.supports(S)
```

would yield

```
0.3 (schalke)
```

We will now give a formal definition of the probabilities of fact goals.

Definition 20 (Probabilities of fact goals):
Fact goals ask for propositions in the db-context. So formally, for a factgoal (atom or negated atom), it is

$$\begin{aligned} P(\varphi) &= p \Longleftrightarrow M \models \mathrm{db}[p^t/p^f/p^i/p^u\ \varphi] \wedge p = p^t \\ P(!\varphi) &= p \Longleftrightarrow M \models \mathrm{db}[p^t/p^f/p^i/p^u\ \varphi] \wedge p = p^f \end{aligned}$$

We apply a strict interpretation, saying that the probability that a proposition is true is the probability that it is positive and not negative, and the other way around for the probability that the negation of the proposition is true. Therefore, p^i is ignored.

Content-oriented Queries The next kind of queries we are going to discuss are content-oriented queries which return the contexts or objects for which the given conditions become true. Such contexts might be documents, annotations or fragments. Content-oriented queries are realised by fact goals as a context subgoal (csubgoal). Consider the following POLAR program:

```
d1[0.4 moral f1||0.8 moral 0.8 *a1|| 0.6 *a2]
a1[0.5 political 0.7 @a3]
a2[0.2 political]
a3[0.6 good 0.5 comment]
```

The query for documents or annotations about 'political'

```
?- D[political]
```

returns

```
0.5 (a1)
0.2 (a2)
```

In case we remember that we annotated a part containing 'moral' and we want to retrieve that fragment, then the query

```
?- F||moral||
```

returns

```
0.8 (f1)
```

Now let us do some knowledge augmentation. The query

```
?- //D[political]
```

returns

```
0.472 (d1)     # from d1(f1,a1,a2)
0.5   (a1)
0.2   (a2)
```

We give a definition of the probabilities of object subgoals.

Definition 21 (Probabilities of context subgoals):
Formally, the probabilities of context subgoals are defined as

$$\begin{aligned}
P(d[\varphi]) &= p \Longleftrightarrow M \models \mathrm{d}[p^t/p^f/p^i/p^u\ \varphi] \wedge p = p^t \\
P(d[!\varphi]) &= p \Longleftrightarrow M \models \mathrm{d}[p^t/p^f/p^i/p^u\ \varphi] \wedge p = p^f \\
P(f||\varphi||) &= p \Longleftrightarrow M \models \mathrm{f}||p^t/p^f/p^i/p^u\ \varphi|| \wedge p = p^t \\
P(f||!\varphi||) &= p \Longleftrightarrow M \models \mathrm{f}||p^t/p^f/p^i/p^u\ \varphi|| \wedge p = p^f \\
P(//d[\varphi]) &= p \Longleftrightarrow M \models \mathrm{d(s1,...,sn)}[p^t/p^f/p^i/p^u\ \varphi] \wedge p = p^t \\
P(//d[!\varphi]) &= p \Longleftrightarrow M \models \mathrm{d(s1,...,sn)}[p^t/p^f/p^i/p^u\ \varphi] \wedge p = p^f
\end{aligned}$$

The last two definitions cover knowledge augmentation. In order to derive d(s1,...,sn), an arbitrary world w is chosen with $R_d(w) \neq \emptyset$. Then, d(s1,...,sn) = createAugmContextExpr(M,d,w,()).

Structure Queries Structure queries ask for all annotations annotating a document, all fragments of a document or all referenced objects. They manifest itself as annorefgoals, referencegoals and fragmentgoals in context subgoals. Taking the above example, the query

```
?- d1[*A]
```

would yield all (positive) content annotations annotating d1, ranked by decreasing access probability:

```
0.8 (a1)
0.6 (a2)
```

In a similar way we could ask for negative content annotations, (positive and negative) meta annotations and referenced objects:

```
?- d1[-*A]    # returns all negative content annotations of d1
?- d1[@A]     # returns all (positive) meta annotations of d1
?- d1[-@A]    # returns all negative meta annotations of d1
?- d1[=>R]    # returns all objects referenced by d1
```

Furthermore, the query

```
?- d1[||F]
```

yields all fragments of d1, again ranked by decreasing access probability. In a similar way, the queries

```
?- f1||*A||
?- f1||-*A||
?- f1||@A||
?- f1||-@A||
```

return all (positive resp. negative) content or meta annotations, respectively, of a fragment f1. On the other hand,

```
?- F||*a1||
?- F||@a1||
```

would return all fragments a1 is (positive) content or meta annotation of, again ranked by the access probability.

The corresponding probabilities are computed as follows.

Definition 22 (Probabilities of annorefs, references and fragments):
The probabilities of annorefs, references and fragments are defined as:

$$\begin{aligned}
P(d[*a]) &= p \Longleftrightarrow M \models_C \text{d[}p\ \text{*a]} \\
P(d[-*a]) &= p \Longleftrightarrow M \models_C \text{d[}p\ \text{-*a]} \\
P(d[@a]) &= p \Longleftrightarrow M \models_C \text{d[}p\ \text{-@a]} \\
P(d[@a]) &= p \Longleftrightarrow M \models_C \text{d[}p\ \text{-@a]} \\
P(d[=>r]) &= p \Longleftrightarrow M \models_C \text{d[}p\ \text{=>r]} \\
P(d[||f]) &= p \Longleftrightarrow M \models_C \text{d[}p\ \text{f||...||]} \\
P(f||*a||) &= p \Longleftrightarrow M \models_C \text{f||}p\ \text{*a||} \\
P(f||-*a||) &= p \Longleftrightarrow M \models_C \text{f||}p\ \text{-*a||} \\
P(f||@a||) &= p \Longleftrightarrow M \models_C \text{f||}p\ \text{@a||} \\
P(f||-@a||) &= p \Longleftrightarrow M \models_C \text{f||}p\ \text{-@a||}
\end{aligned}$$

We apply context-validity (see Def. 13) to ensure that the actual structure established by the subcontexts is reflected.

Complex Queries Complex queries consist of a subgoal list which forms a conjunction of subgoals. We regard subgoals as being independent, so the final probability is the product of the probabilities of each subgoal or context subgoal. As an example for lists of context subgoals, take

```
?- D[good & comment]
```

which returns

```
0.3 (a3)
```

which is the product of the probabilities of both terms in a3 ($0.5 \cdot 0.6$). As a further example, take

```
?- D[political & @A] & A[good & comment]
```

which would ask about documents about 'political' which are also told to be good comments. This query would return

```
0.105 (a1)
```

because $0.105 = 0.5 \cdot 0.7 \cdot 0.6 \cdot 0.5$. An equivalent query would be

```
?- D[political] & D[@A] & A[good] & A[comment]
```

which leads to the same result.

Having said that we assume all subgoals to be independent, there are exceptions. Consider the query

```
?- D[t1 & !t1]
```

There are two possible strategies handling this: return the probability that t1 is *inconsistent* in a context, or return nothing. The latter interpretation would declare t1 and !t1 as disjoint events. Which strategy to choose and how to handle such a situation is subject to the actual implementation (e.g. by applying an intensional evaluation) and shall not be elaborated further at this point.

5.2.4.2 Rules

After having discussed queries, we are now going to discuss POLAR rules. POLAR rules consist of a head and, like queries, a corresponding subgoal list. Note that we do not support probabilistic rules in POLAR yet.

In POLAR, rules like

```
document(D) :- book(D)
```

to capture dependencies or generate new knowledge in the database context db or

```
D[european_city(C)] :- D[german_city(C)]
```

for generating knowledge or describing dependencies within contexts are supported. Another kind of rules generate new global knowledge or describe dependencies between objects in different contexts, for example

```
relevant(D) :- D[@A] &  A[good & introduction]
```

If we recall the set notation of the four truth values, the rules above have in common that their goal (or head) is positive. But POLAR also supports rules which generate negative knowledge, like

```
!relevant(D) :- D[@A] &  A[bad & introduction]
```

Positive and negative intensional knowledge can then be used to generate the four truth values for the derived proposition in a context, as we will see later. But first we have to discuss how POLAR rules are evaluated.

Evaluation of Rules From Fuhr (2000) we borrow the notion of *event keys* and *event expressions*. An event is any fact in a context, but also the reference to a (content or meta) annotation or another object. Let ε be a function which assigns an event key to each instantiated atom, annoref, reference in a context. For example, let

$$\begin{aligned} \varepsilon(\text{d1 [*a1]}) &= \texttt{d1[}\star\texttt{a1]} \\ \varepsilon(\text{t1}) &= \texttt{t1} \end{aligned}$$

etc. Based on event keys, we define event expressions:

Definition 23 (Event expressions):
An *event expression* is a Boolean expression and recursively defined by a function η as follows:

1. $\eta(L) := \bigvee_s \eta(s)$ where s is a rule whose head matches L.
2. $\eta(L_0 :- L_1 \& \ldots \& L_n) := \eta(L_1) \wedge \ldots \wedge \eta(L_n)$
3. $\eta(c[L_1 \& \ldots \& L_n]) := \eta(c[L_1]) \wedge \ldots \wedge \eta(c[L_n])$ if c is a context name
4. $\eta(g) = \varepsilon(g)$ if g is a proposition, annoref or reference in a context. g is called an *event atom*.

An example shall illustrate event expressions.

Example 10 (Event expressions): Consider the following simple POLAR program:

```
d1[ 0.6 ir  0.8 db   @a1   @a2]
a1[ 0.6 good   0.5 introduction ]
a2[ 0.4 bad  0.7 introduction ]
relevant1(D) :- D[ir] & D[@A] & A[good & introduction]
!relevant1(D) :- D[@A] & A[bad & introduction]
relevant2(D) :- D[ir]
relevant2(D) :- D[db]
relevant3(D) :- D[@A] & A[introduction]
```

For relevant1(d1), the event expression is

η(relevant1(d1)) =
η(relevant1(d1) :- d1[ir & @a1] & a1[good & introduction]) =
    η(d1[ir & @a1] $\wedge$ η(a1[good & introduction]) =
    η(d1[ir] $\wedge$ η(d1[@a1]) $\wedge$ η(a1[good]) $\wedge$ η(a1[introduction]) =
    d1[ir] $\wedge$ d1[@a1] $\wedge$ a1[good] $\wedge$ a1[introduction]

For !relevant1(d1), the event expression is

η(!relevant1(d1)) =
η(!relevant1(d1) :- d1[@a2] & a2[bad & introduction]) =
η(d1[@a2] $\wedge$ η(a2[bad & introduction]) =
η(d1[@a2]) $\wedge$ $\wedge\eta$(a2[bad]) $\wedge$ η(a2[introduction]) =
`d1[@a2] ∧ a2[bad] ∧ a2[introduction]`

For relevant2(d1), the event expression is

η(relevant2(d1)) =
η(relevant2(d1) :- d1[ir]) $\vee$ η(relevant2(d1) :- d1[db]) =
η(d1[ir]) $\vee$ η(d1[db]) =
`d1[ir] ∨ d1[db]`

For relevant3(d1), the event expression is

η(relevant3(d1)) =
η(relevant3(d1) :- d1[@a1] & a1[introduction]) $\vee$
η(relevant3(d1) :- d1[@a2] & a2[introduction]) =
η(d1[@a1]) $\wedge$ η(a1[introduction]) $\vee$ η(d1[@a2]) $\wedge$ η(a2[introduction]) =
`d1[@a1] ∧ a1[introduction] ∨ d1[@a2] ∧ a2[introduction]`

□

Event expressions are Boolean expressions and can thus be transformed into disjunctive normal form (DNF) like $K_1 \vee \ldots \vee K_n$, where each K_i is an event atom or a conjunction of event atoms. We will now describe how we calculate the probabilities of our newly derived facts, which is the probability of the corresponding event expressions. For event atoms, we apply definitions 20, 21 and 22. This means that we ignore inconsistent knowledge for deducing new knowledge, but we may induce new inconsistent knowledge.

Probabilities of Event Expressions We assume the independence of event atoms, so the probability of their conjunction is calculated as the product of their probability, i.e. $P(K_i) = P(a_1) \cdot \ldots \cdot P(a_n)$ if $K_i = a_1 \wedge \ldots \wedge a_n$ (an exception are disjoint events like $a \wedge !a$). In order to compute the probability of the disjunction, we apply the inclusion-exclusion formula (sometimes also known as the *sieve formula* (Storch and Wiebe, 1989, p. 159)).

Definition 24 (Inclusion-exclusion formula):
Let $K_1 \vee \ldots \vee K_n$ be an event expression in DNF. The probability of this expression is computed with the *inclusion-exclusion formula*

$$P(K_1 \vee \ldots \vee K_n) = \sum_{i=1}^{n} (-1)^{i-1} \left(\sum_{\substack{1 \leq j_1 < \\ \ldots < j_i \leq n}} P(K_{j_1} \wedge \ldots \wedge K_{j_i}) \right)$$

and

$$P(K_{j_1} \wedge \ldots \wedge K_{j_i}) = P(K_{j_1}) \cdot \ldots \cdot P(K_{j_i})$$

For example, for the expression $K_1 \vee K_2$ we obtain:

$$P(K_1) + P(K_2) - P(K_1) \cdot P(K_2).$$

Example 11 (Probabilities of event expressions): We recall Example 10 and the event expressions in this example. We obtain for relevant1(d1):

$P(\text{relevant1(d1)}) =$
$P(\texttt{d1[ir]} \wedge \texttt{d1[@a1]} \wedge \texttt{a1[good]} \wedge \texttt{a1[introduction]}) =$
$0.6 \cdot 1 \cdot 0.6 \cdot 0.5 = 0.18$

For !relevant1(d1):

$P(\text{!relevant1(d1)}) =$
$P(\texttt{d1[@a2]} \wedge \texttt{a2[bad]} \wedge \texttt{a2[introduction]}) =$
$P(\texttt{d1[@a2]}) \cdot P(\texttt{a2[bad]}) \cdot P(\texttt{a2[introduction]}) = 1 \cdot 0.4 \cdot 0.7 = 0.28$

For relevant2(d1):

$P(\text{relevant2(d1)}) =$
$P(\texttt{d1[ir]} \vee \texttt{d1[db]}) =$
$P(\texttt{d1[ir]}) + P(\texttt{d1[db]}) - P(\texttt{d1[ir]}) \cdot P(\texttt{d1[db]}) =$
$0.6 + 0.8 - 0.6 \cdot 0.8 = 0.92$

For relevant3(d1):

$P(\text{relevant3(d1)}) =$
$P(\texttt{d1[@a1]} \wedge \texttt{a1[introduction]} \vee \texttt{d1[@a2]} \wedge \texttt{a2[introduction]}) =$
$P(\texttt{d1[@a1]} \wedge \texttt{a1[introduction]}) + P(\texttt{d1[@a2]} \wedge \texttt{a2[introduction]}) -$
$P(\texttt{d1[@a1]} \wedge \texttt{a1[introduction]}) \cdot$
$P(\texttt{d1[@a2]} \wedge \texttt{a2[introduction]}) =$
$P(\texttt{d1[@a1]}) \cdot P(\texttt{a1[introduction]}) +$
$P(\texttt{d1[@a2]}) \cdot P(\texttt{a2[introduction]}) -$
$P(\texttt{d1[@a1]}) \cdot P(\texttt{a1[introduction]}) \cdot$
$P(\texttt{d1[@a2]}) \cdot P(\texttt{a2[introduction]}) =$
$1 \cdot 0.5 + 1 \cdot 0.7 - 1 \cdot 0.5 \cdot 1 \cdot 0.7 = 0.5 + 0.7 - 0.5 \cdot 0.7 = 0.85$

□

Model of a POLAR program In Definition 15 we defined the model of a basic POLAR program. Basic POLAR programs do not contain rules, and we now want to show how a model of full POLAR programs must look like.

We have shown how to calculate probabilities for positive and negative rules by means of event expressions. Positive rules calculate the probability that an instantiated head is *true* or *inconsistent*, whereas negative rules compute the probability that an instantiated head is *false* or *inconsistent*. If we combine these values, we gain the probabilities for the four truth values for and instantiated head. An example shall illustrate this.

Example 12 (Probabilities of instantiated rule heads): Based on Example 11, we calculate the probabilities for relevant1(d1):

$$\begin{aligned} p_t &= 0.18 \cdot (1 - 0.28) = 0.1296 \\ p_f &= 0.28 \cdot (1 - 0.18) = 0.2296 \\ p_i &= 0.18 \cdot 0.28 = 0.0504 \\ p_u &= 1 - 0.18 \cdot 1 - (0.28) = 0.5904 \end{aligned}$$

p_t is the probability that relevant1(d1) is positive and not negative. p_f is the probability that relevant1(d1) is negative and not positive. p_i is the probability that relevant1(d1) is positive and negative. Finally, p_u is the probability that relevant1(d1) is neither positive nor negative. So we get db[0.1296/0.2296/0.0504/0.5904 relevant1(d1)]. □

The probabilities of newly derived knowledge depends on the probabilities calculated from the corresponding event expressions. We formulate this as a constraint:

Constraint 13 (Probabilities of instantiated rule heads): For each instantiation of rule heads, the following condition must hold:

$$\begin{aligned} M \models \mathrm{d}[p_t/p_f/p_i/p_u\ \varphi] \iff\ & p_t = P(\eta(\mathrm{d}[\varphi])) \cdot (1 - P(\eta(\mathrm{d}[!\varphi])) \\ & p_f = P(\eta(\mathrm{d}[!\varphi])) \cdot (1 - P(\eta(\mathrm{d}[\varphi])) \\ & p_i = P(\eta(\mathrm{d}[\varphi])) \cdot P(\eta(\mathrm{d}[!\varphi]) \\ & p_u = (1 - P(\eta(\mathrm{d}[\varphi]))) \cdot (1 - P(\eta(\mathrm{d}[!\varphi])) \end{aligned}$$

If the instantiated head is true w.r.t. the interpretation structure M, then the corresponding probabilities must be derived from the subgoal list and their event expressions. On the other hand, if we derive the probabilities from the subgoal list, then the instantiated head must be true w.r.t. M with these probabilities. □

With this constraint, we now define a model for a full POLAR program.

Definition 25 (Model of a POLAR program):
An interpretation structure M is a *model* of a POLAR program P iff

- definition 15 is applied and additionally
- constraint 13 holds.

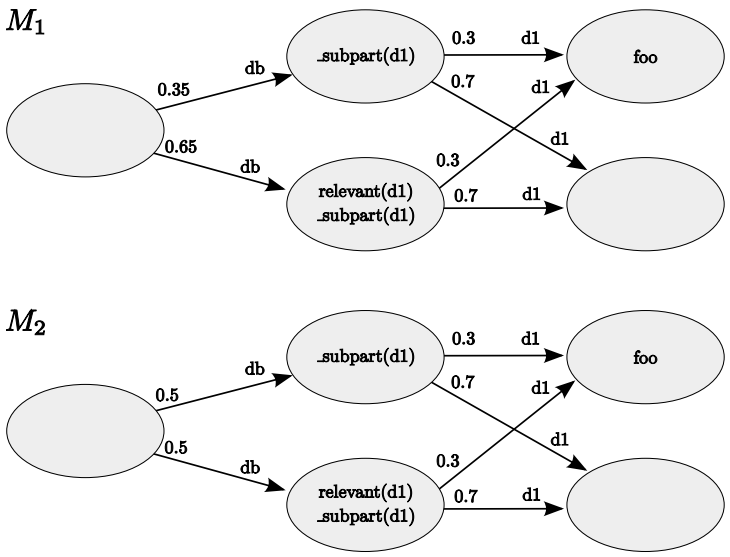


Figure 5.16: Two interpretation structures for the POLAR model example

Example 13 (Model of a POLAR program): Consider the POLAR program

```
0.5 relevant(d1)
d1[ 0.3 foo ]
relevant(D) :- D[foo]
```

The event expression for "relevant(d1)" is `relevant(d1)` $\vee$ `d1(foo]`; the probability of this expression is $0.5 + 0.3 - 0.5 \cdot 0.3 = 0.65$. Figure 5.16 shows two interpretation structures. M_1 is a model of the POLAR program above, while M_2 is only a model of the basic POLAR program, consisting of the first two lines, but not of the whole program due to different probabilities of the worlds reachable by contextagent db. For the whole program, constraint 13 is violated in M_2. □

5.3 Retrieval Function

In the last section we have discussed knowledge modelling in POLAR, including knowledge augmentation, and how queries and rules are processed and interpreted semantically. This allows for queries about the content of documents and annotations, and complex queries and rules where knowledge from annotations and annotated objects can be combined. What is still missing is the introduction of a "real" retrieval function which includes and combines well-known concepts like term spaces based on measures like the inverse document frequency, within-context term weights and probabilistic and logic-based IR. As in POOL, retrieval functions in POLAR are based on probabilistic inference.

5.3.1 Information Retrieval with Probabilistic Inference

Probabilistic IR computes the probability that a document is relevant w.r.t. a query q and a document d, which is denoted $P(R|d,q)$. Van Rijsbergen coined the paradigm of IR as uncertain inference where the probability $P(d \rightarrow q)$ that a document d implies a query q, is estimated (van Rijsbergen, 1986). This probability is assumed to capture the notion of relevance, i.e. it

is proportional to $P(R|d,q)$, so if we are not interested in the exact probabilities $P(R|d,q)$, the same ranking can be achieved by calculating $P(d \rightarrow q)$. In van Rijsbergen (1986), a definition of $P(d \rightarrow q)$ is given as

$$P(d \rightarrow q) := P(q|d).$$

Based on this, Wong and Yao (1995) show that many popular retrieval functions can be modelled in terms of probabilistic inference. For example, the vector space model can be expressed as uncertain inference:

$$P(d \rightarrow q) = P(q|d) = \sum_t P(t|d) \cdot P(q|t) = \sum_t P(d \rightarrow t) \cdot P(t \rightarrow q) = \vec{d} \cdot \vec{q} \tag{5.3}$$

given that all t are disjoint, i.e. $P(t_i \wedge t_j) = 0$ if $i \neq j$ and $\sum_t P(t) = 1$. $\vec{d} \cdot \vec{q}$ calculates the scalar product of the document and query vector. $P(d \rightarrow t)$ can be considered as a representation of the document concepts and $P(t \rightarrow q)$ as one of the query concepts. As argued in Wong and Yao (1995), indexing estimates the degree to which a document is relevant to a concept, whereas query formulation goes the other way round and describes the degree to which a concept is relevant to a query.

5.3.2 Probabilistic Inference in POLAR

5.3.2.1 Syntax

POLAR supports retrieval based on probabilistic inference. To do so, we model queries as contexts. For example,

```
q1[ 0.6 is  0.9 db ]
```

says that the query q1 consists of the query terms "is" and "db". By modelling queries as contexts, we are not only able to specify query term weights (in this case 0.6 and 0.9, respectively), but also to formulate queries containing categorisations and attributes. The expression

```
?- D->q
```

retrieves all instantiations d of the variable `D` and their corresponding probability $P(d \rightarrow q)$. Note that there is another advantage of modelling queries as contexts: $P(d \rightarrow q)$ calculates the implication between contexts in general (called *context implication* (Rölleke, 2008)), so it is not only defined between a document and a query, but could also be used, e.g., to compute the similarity between documents.

5.3.2.2 Semantics

In principle, probabilistic inference in POLAR is the same as in POOL. This is possible because in POLAR, although we calculate augmented values differently and introduce special propositions and polarity, we are using the same semantic interpretation structures as in POOL (namely Kripke structures). This means that we can exploit POOL's probabilistic inference as described in (Rölleke, 1998, chapter 4) for POLAR. As indicated in Wong and Yao (1995), there can potentially be more than one way to estimate the implication probability for a document and a query. In (Rölleke, 1998, chapter 4) we see how the vector space model can be described by means of our probabilistic interpretation structure. This can be done using the combination of fact, query and predicate spaces – the fact space is established by the propositions in a context, the query space by the propositions in the query, and the *predicate space* is a new

space which reflects the importance of a certain predicate. A predicate can be a term predicate, a category predicate or an attribute predicate. A typical example of a term predicate space is a space consisting of terms and their associated *inverse document frequencies*[5]. Category predicate spaces consist of category names and their associated probabilities. Analogously, attribute spaces contain attribute names and their probabilities. Similar to the probabilities in term spaces, category and attribute spaces reflect the importance of a category or an attribute. This importance value can again be calculated by means of an inverse document frequency (depending on the number of documents a category/attribute name appears in) or directly given by the user.

We do not go into more detail regarding the semantics of the POLAR expression "`D->q`" here. POLAR is supposed to support different interpretations of $P(d \rightarrow q)$, and to pick and thoroughly explain a certain one would go too far here. Instead, we refer to (Rölleke, 1998, chapter 4) for a more exhaustive description on and general proof of concept of how the implication probability can be interpreted semantically (in that particular case by means of probabilistic inference modelling the vector space model with Eq. 5.3). The method explained there for POOL can be applied to POLAR as well. Additionally, we refer to Section 6.2.4 where we further discuss probabilistic inference and possible retrieval functions when introducing the implementation of POLAR.

5.3.2.3 Predicate Spaces in POLAR

Although the predicate spaces might be extracted from the collection statistics, POLAR offers syntactic expressions to define them explicitly (see Fig. 5.1 on page 90). The expression

```
0.5 °databases
```

sets $P(\text{databases}) = 0.5$ in the term space.

```
0.7 °°footballclub
```

does the same for class "footballclub" in the category space and

```
0.8 °°°author
```

sets the probability of the attribute "author" in the attribute space to 0.8.

5.4 Summary and Discussion

In this section, we defined the syntax and semantics of POLAR, together with the underlying annotation-based retrieval model. With POLAR, we can model and query structured annotation hypertexts.

As an extension to POOL, the POLAR syntax allows for specifying contexts and subcontexts, according to a given type (subpart, merged annotation target, fragment, (negative) content and meta annotation, and reference). Access probabilities to the subcontexts can be given. Annotations, documents and subparts are contexts containing propositions which can have four truth values, *true*, *false*, *inconsistent* and *unknown*. Complex queries to the underlying structured annotation hypertext are supported in order to satisfy advanced information needs. Rules can be used to deduce new knowledge and to support queries.

[5] Similar to an n-dimensional vector space: each dimension reflects a term in the collection; a term space vector then represents the distribution of inverse document frequencies over the terms.

The semantics of POLAR are described by a probabilistic variant of Kripke structures based on possible worlds, which are known from modal logic. We described the semantics of basic knowledge modelling and the constraints which make an interpretation structure a model of a basic POLAR program. If propositions are made in a context, this means that the corresponding contextagent reaches worlds where these propositions are contained. The probability of a proposition to be *true*, *false*, *inconsistent* or *unknown* is determined by the probabilities with which worlds containing the proposition are accessed. Access to subcontexts (annotations, fragments, targets, referenced objects, subparts) is represented by accessibility relations and the worlds a contextagent can reach from the current one. These accessibility relations determine what the contextagent believes to be possible, and they reflect the behaviour of a user accessing annotations or other subcontexts (and believing their respective assertions) with a certain probability. To make a distinction between the different kinds of subcontexts in POLAR, special propositions for the relation type were introduced. Constraints are introduced which need to be applied in order to make an implementation structure a model of a basic POLAR program.

Furthermore, we discussed how knowledge augmentation, one of the core concepts behind POLAR, works. Knowledge augmentation simulates the user accessing subcontexts and aggregates the knowledge a user would gain when traversing the possible worlds. This aggregated knowledge is again contained in possible worlds which an augmented context would access with a certain probability. These possible worlds and its content are derived by an extension of POOL's *trees* function (which also takes the polarity of annotations into account) and a modification of the truth value assignment function. In order to take certain peculiarities of the different subcontext types in POLAR into account, the notion of valid augmented context expressions is introduced and an algorithm is discussed which derives such expressions from the interpretation structure. Such valid augmented context expressions control the creation of the augmented subcontexts.

We further introduced how queries and rules are processed. For rules, we applied the notion of event expressions to deduce new knowledge. This led to an additional constraint which needs to be adhered to in order to make an interpretation structure a model of a full POLAR program.

Finally, we sketched POLAR's retrieval function which is based on the notion of uncertain inference, where $P(d \rightarrow q)$, the probability that a document d implies a query q, is estimated. Since many interpretations of such a function are conceivable (for example a realisation of the vector space model, as it was done in POOL), we did not restrict ourselves to only one of them, but just presented predicate spaces which should be incorporated in possible POLAR interpretations of $P(d \rightarrow q)$. A precise implementation of $P(d \rightarrow q)$ is subject to the discussion in the next chapter.

The semantics of POOL and POLAR basically share the same interpretation structure. As mentioned in the last chapter, POLAR is in a way a generalisation of POOL (every POOL program is also a POLAR program) or a specialisation (POLAR specialises relation types and offers additional functionality, a higher expressiveness, typification of contexts and relations, which was not an issue for POOL. POOL relies on a simpler model.). This is also reflected in the semantics, where we introduced special propositions in order to distinguish between relation and subcontext types in POLAR, and adapted POOL's original *trees* function in order to deal with the peculiarities of the different types. We also need to handle possible cycles in POLAR's interpretation structure when performing knowledge augmentation.

To model different relation types, we used special propositions within worlds. An alternative option would have been to distinguish between different relation types directly. For example, we could define a set $R_{a_i}^{canno} \subseteq R_{a_i}$ which contains only content annotation relations. The problem here is that we need to extend the definition of our interpretation structure with syntactic elements, so we cannot keep it as simple as possible, as it would be required to keep a clear distinction between syntax and semantics. On the other hand, what does, e.g., `d1[*a1]` mean? It means that contextagent d1 knows that a1 is an annotation (or, in other words, it knows that a1 would reach another world through an annotation, at least w.r.t. context d1), so this is certainly knowledge a contextagent has in a specific world. It is therefore straightforward to assign this knowledge to its corresponding worlds, without further expanding the interpretation structure. The advantage is that we can keep the semantic structure simple and no syntactic constructs need to make their way into the definition of the interpretation structure. The information about the relation types moves from the interpretation structure itself – as it would be when applying the first option – to an instance of the interpretation structure. This way, POLAR's interpretation structure stays compatible to the one of POOL.

Having specified POLAR's syntax and semantics in this chapter, the next question is how POLAR can be implemented. In the subsequent chapter, a possible implementation based on four-valued probabilistic Datalog is discussed.

6

POLAR Implementation

Logic, like whiskey, loses its beneficial effect when taken in too large quantities.

(Lord Dunsany)

In the last chapter we discussed POLAR's syntax and semantics. We shed some light on one of POLAR's core concepts, knowledge augmentation, and besides POLAR's knowledge modelling abilities, we also looked at retrieval functions based on probabilistic inference. The aim of this chapter is to introduce a possible implementation of POLAR, which is based on four-valued probabilistic Datalog (FVPD). FVPD supports four-valued logics as we know it from POLAR, as well as the open-world assumption. With HySpirit[1], there exists an implementation of FVPD which can be used as an engine to execute POLAR programs once we defined an appropriate translation from POLAR to FVPD.

The chapter is structured as follows. First, four-valued probabilistic Datalog is introduced which also discusses its translation into probabilistic Datalog. Subsequently, the translation of POLAR programs into FVPD is described for basic knowledge modelling, queries and rules, augmentation and possible retrieval functions.

6.1 Four-Valued Probabilistic Datalog (FVPD)

Four-valued probabilistic Datalog (FVPD) is an extension of probabilistic Datalog. Similar to Prolog, its syntax consists of variables, constants, predicates and Horn clauses. Probabilities can be assigned to facts. Semantically, FVPD uses the four different truth values we already know from POLAR, *true*, *false*, *inconsistent* and *unknown* (Fuhr and Rölleke, 1998; Rölleke and Fuhr, 1996). Therefore, FVPD can deal with the open world assumption – facts which cannot be deduced are not assumed as false, as it would be with a closed world assumption. This makes FVPD an excellent candidate for the implementation of POLAR. We begin with a brief description of FVPD's syntax. As a extension of probabilistic Datalog, FVPD can be mapped onto Datalog rules and facts. This translation is described in Section 6.1.2, which also contains an example of an FVPD program and how it is translated and evaluated.

[1] `http://qmir.dcs.qmul.ac.uk/hyspirit.php`

6.1.1 Syntax of FVPD

Facts, Rules and Queries We introduce the syntax of FVPD as described in Fuhr and Rölleke (1998). Basic elements in FVPD are *predicates* (alphanumeric strings starting with lower-case letters; sometimes also referred to as *relations*), *constants* (numbers or alphanumeric strings starting with lower-case letters) and *variables* (alphanumeric strings starting with upper-case letters). A *term* is either a constant or a variable. A *ground term* is a constant. The *Herbrand Universe* of a FVPD program is the set of constants occurring in it. An *atom* $q(t_1, \ldots, t_n)$ consists of the n-ary predicate symbol q and a list of arguments $(t_1, \ldots, t_n)$ with each t_i being a term. A *literal* is an atom $q(t_1, \ldots, t_n)$ or a negated atom $!q(t_1, \ldots, t_n)$. A *ground fact* is a literal where all arguments are constants. Ground facts may be preceded by a *weight list*, which is equal to a POLAR weight list discussed in Sections 5.1.1 and 5.2.1.2. *Rules* consist of a head and a subgoal list. *Subgoal lists* are conjunctions of subgoals; each *subgoal* is a literal. A *head* is a literal with the restriction that all variables appearing in the head must also appear in the corresponding subgoal list. *Queries* are headless subgoal lists preceded by "?-".

Facts describe the *extensional* knowledge of an FVPD program. By applying rules, new *intensional* facts are created.

Closed World Assumption and Disjointness Keys In FVPD it is possible to assume a closed world for whole predicates and define the disjointness of events. In order to define certain tuples of a relation as disjoint events, a *disjointness key* can be declared for a predicate. For example, the expression `_dk(termspace, "$1")` says that all tuples of the predicate called `termspace` which are equal in the first argument should are regarded as being disjoint[2]. In FVPD, the expression `_CWA(termspace)` would declare `termspace` as being a two-valued predicate with an underlying closed world assumption. In contrast to that, the expressions `_OWA(term)` and `_dk(term, "$1,$2")` would declare `term` as being four-valued with an underlying open world assumption; all tuples of this relation are disjoint if they are equal in the first and second argument (see also Example 14 below). If the disjointness key is omitted for a predicate/relation, we assume that it does not contain any disjoint event.

6.1.2 Translation to and Evaluation with Probabilistic Datalog

FVPD is built upon *probabilistic Datalog* (PD). Each FVPD program is thus translated into a PD program, which is executed subsequently. We do not give a formal introduction to PD here, but refer to Fuhr (2000) for an exhaustive discussion of PD. Instead, we introduce PD as needed by giving an example. We start with the translation of FVPD facts into PD and continue with rules and queries. Afterwards we present an example FVPD program, its translation into PD and how the resulting PD program is evaluated.

6.1.2.1 Translation of FVPD Facts

FVPD facts are mapped onto up to 3 PD facts (one for representing a truth value). Each FVPD fact

$$p_t/p_f/p_i \;\; q(a_1, \ldots, a_n)$$

[2]To specify a closed or open world for predicates and to define the disjointness key, we use the syntax as provided by current HySpirit implementations, which is slightly different to the one introduced in Fuhr and Rölleke (1998), where the equivalent expression would be `#termspace(dk)`.

where $p_t/p_f/p_i$ are the probabilities for *true*, *false* and *inconsistent* is translated into three two-valued PD facts

$$p_t\, q(t, a_1, \ldots, a_n). \quad p_f\, q(f, a_1, \ldots, a_n). \quad p_i\, q(i, a_1, \ldots, a_n).$$

Each of these facts reflects one truth value. The truth value for *unknown* is implicitly derived as $1 - (p_t + p_f + p_i)$. The PD relations are in turn used in PD rules representing positive and negative evidence:

$$\begin{aligned} pos\_q(a_1, \ldots, a_k) &\text{ :- } q(t, a_1, \ldots, a_k) \\ pos\_q(a_1, \ldots, a_k) &\text{ :- } q(i, a_1, \ldots, a_k) \\ neg\_q(a_1, \ldots, a_k) &\text{ :- } q(f, a_1, \ldots, a_k) \\ neg\_q(a_1, \ldots, a_k) &\text{ :- } q(i, a_1, \ldots, a_k) \end{aligned}$$

This way, $pos\_q(a_1, \ldots, a_k)$ combines all evidence that $q(a_1, \ldots, a_k)$ is positive, which is the case if the fact is *true* or *inconsistent*. Similarly for $neg\_q(a_1, \ldots, a_k)$, which combines all evidence that $q(a_1, \ldots, a_k)$ is negative. Note the analogy to the set notation of the four truth values (see p. 57). Here, a proposition was positive if t appeared in the set notation of its truth value ($\{t, f\}$ for *inconsistent* and $\{t\}$ for *true*). Analogously for negative propositions.

6.1.2.2 Translation of FVPD Rules and Queries

A FVPD rule has the form

$$h \text{ :- } b_1 \,\&\, \ldots \,\&\, b_n$$

and consists of a head h and a conjunction of subgoals $b_1, \ldots, b_n$. The head h is mapped onto the corresponding positive or negative literal in PD, depending on whether h is positive or negative. Each positive subgoal in FVPD is replaced by a conjunction of the corresponding positive PD literal and the negation of the corresponding negative PD literal (and vice versa for negative FVPD subgoals). So for each FVPD rule like above, we gain the PD rule

$$g \text{ :- } r_1 \,\&\, \ldots \,\&\, r_n$$

with

$$g = \begin{cases} pos\_h & \text{if } h \text{ is positive,} \\ neg\_h' & \text{if } h = !h' \text{ (h is negative),} \end{cases}$$

and

$$r_i = \begin{cases} pos\_b_i \;\&\; !neg\_b_i & \text{if } b_i \text{ is a positive literal,} \\ neg\_b_i' \;\&\; !pos\_b_i' & \text{if } b_i' = !b_i \text{ is a positive literal.} \end{cases}$$

6.1.2.3 Example

We illustrate the translation of FVPD programs and their evaluation in PD with an example.

Example 14 (FVPD translation and evaluation): Consider the following FVPD program with facts, rules and a query:

```
1  _CWA(acc_canno)
2  _dk(term,"$1,$2")
3  0.5 acc_canno(d1,a1).
4  0.6/0.1 term(football,a1).
5  term_k(T,D) :- term(T,D).
6  term_k(T,D) :- acc_canno(D,S) & term_k(T,S).
7  ?- term_k(football,D)
```

The first line defines `acc_canno` as two-valued relation (closed-world assumption). The second line declares the tuples of the relation `term` as independent. The FVPD program would be translated into the following PD program:

```
1  _dk(term,"$2,$3")
2  0.5 acc_canno(d1,a1).
3  0.6 term(t,football,a1).
4  0.1 term(f,football,a1).
5  pos_term(T,D) :- term(t,T,D).
6  pos_term(T,D) :- term(i,T,D).
7  neg_term(T,D) :- term(f,T,D).
8  neg_term(T,D) :- term(i,T,D).
9  pos_term_k(T,D) :- pos_term(T,D) & !neg_term(T,D).
10 neg_term_k(T,D) :- neg_term(T,D) & !pos_term(T,D).
11 pos_term_k(T,D) :- acc_canno(D,S) &
12                    pos_term_k(T,S) & !neg_term_k(T,S).
13 neg_term_k(T,D) :- acc_canno(D,S) &
14                    neg_term_k(T,S) & !pos_term_k(T,S).
15 ?- pos_term_k(football,D) & !neg_term_k(football,D)
```

The first line in the PD listing translates the disjointness key of the second line in the FVPD listing. In the PD `term` relation, the 2nd and 3rd attribute (which represent the 1st and 2nd attribute in the corresponding FVPD relation) are independent – the disjointness key spans over these two attributes. If two tuples in the `term` relation are equal regarding the 2nd and 3rd attribute, they are disjoint w.r.t. the first attribute. This way it is ensured that the four truth values are disjoint as required. For example, in lines 3 and 4 we see the translation of `term(football,a1)`; here, `term(t,football,a1)` and `term(f,football,a1)` are disjoint events, since the second and third attribute are the same. Line 2 is the PD expression of the `acc_canno` fact. Since it is declared as being two-valued in the FVPD listing, it remains unchanged in the translation. Lines 5 – 8 declare the PD relations for positive and negative evidence w.r.t. `term`. Lines 9 to 14 are the translation of the rules, and the last line is the PD version of the query.

We now show an example how the FVPD query

```
?- term_k(football,D).
```

which is translated into the PD query

```
?- pos_term_k(football,D) & !neg_term_k(football,D).
```

is evaluated for d1 and a1. After the FVPD query has been translated into PD, a PD engine like HySpirit computes the corresponding probabilities

$$P(\text{pos\_term\_k(football,d1) \& !neg\_term\_k(football,d1)})$$

and

$$P(\text{pos\_term\_k(football,a1) \& !neg\_term\_k(football,a1)})$$

and returns them as the final result. For the evaluation of the query, the PD engine utilises a function η, which evaluates PD expressions and returns PD event expressions (similar to Def. 23, where we defined POLAR event expressions), and the inclusion-exclusion-formula defined in Def. 24 on page 133. See Fuhr (2000) for further details. During evaluation of a possible instantiation of the variables, the query is transformed into a Boolean expression consisting of basic event expressions for which the probabilities are known and whose combined probability is calculated with the inclusion-exclusion formula. In Appendix B.2, we show that

$$\eta(\text{pos\_term\_k(football,d1) \& !neg\_term\_k(football,d1)}) = \texttt{acc\_canno(d1,a1)} \wedge \texttt{term(t,football,a1)}$$

due to the application of rules 11 – 14 (which in turn invoke rules 5 to 10) and the disjointness key of the `term` relation. We arrive at

$$P(\text{pos\_term\_k(football,d1) \& !neg\_term\_k(football,d1)}) = P(\texttt{acc\_canno(d1,a1)} \wedge \texttt{term(t,football,a1)}) = 0.5 \cdot 0.6 = 0.3.$$

In a similar way, we can show that

$$P(\text{pos\_term\_k(football,a1) \& !neg\_term\_k(football,a1)}) = P(\texttt{term(t,football,a1)}) = 0.6$$

due to the rules in lines 9 and 10. So our FVPD query "`?- term_k(football,D)`" yields

```
0.6 (a1)
0.3 (d1)
```

□

We have introduced four-valued probabilistic Datalog (FVPD), which is an extension of probabilistic Datalog capable of dealing with four truth values and an open world assumption. We presented the syntax of FVPD and its translation into probabilistic Datalog (PD). PD programs, in turn, can be translated into a probabilistic relational algebra (PRA) (Fuhr and Rölleke, 1997). HySpirit, as an implementation of FVPD, does exactly this – it translates an FVPD program into a PD one and this again into a PRA program. The PRA program is then executed by a corresponding engine provided by HySpirit. The PRA engine conducts the intensional analysis of the event expressions required for calculations such as the one shown in the example above, considering disjointness of events. If we find a correct translation function of POLAR programs to FVPD, we are done – POLAR programs would be translated into PRA ones at the end of the chain and then be executed by HySpirit's PRA engine.

6.2 POLAR Translation to FVPD

POLAR programs are executed by translating them into FVPD and running these programs with the HySpirit engine. To translate POLAR programs, we use a function *trans* which is going to be described in the following. *trans* takes 2 or 3 parameters. The first parameter is

an expression or a set of expressions. The second one is the context of the expressions. The optional last parameter is a flag which, if set to 1, says that knowledge augmentation should be performed. 0 means no knowledge augmentation. We omit this parameter if it is not further needed. We commence with the translation of programs and clauses and go deeper into the translation routines in the following sections. We refer to the description of the POLAR syntax in figures 5.1 and 5.2 on page 90 and 92, respectively.

$$
\begin{aligned}
\mathit{trans}(\text{program,c}) &:= \mathit{trans}(\text{clause,c})\ \mathit{trans}(\text{program',c}) \\
&\quad \text{if program = clause program'} \\
\mathit{trans}(\text{clause,c}) &:= \mathit{trans}(\text{fact,c}) \\
&\quad \text{if clause = fact} \\
&:= \mathit{trans}(\text{query,d}) \\
&\quad \text{if (clause = query)} \\
&:= \mathit{trans}(\text{rule,d}) \\
&\quad \text{if (clause = rule)} \\
&:= \mathit{trans}(\text{context,d}) \\
&\quad \text{if (clause = context)} \\
&:= \mathit{trans}(\text{predicate}) \\
&\quad \text{if (clause = predicate)}
\end{aligned}
$$

A program consists of clauses, so the translation of a program is actually the translation of each clause. A clause can be a fact, query, rule, context or predicate. The next section about basic knowledge modelling describes the translation of fact and context clauses. Section 6.2.2 discusses the translation of queries and rules. Section 6.2.4 contains a discussion about the translation of predicates.

6.2.1 Basic Knowledge Modelling

This section shows the translation of POLAR basic knowledge modelling into FVPD. We start with a few translation examples of POLAR programs known from previous chapters. Afterwards, the corresponding translation rules are discussed.

6.2.1.1 Examples

Example 15 (Translation of POLAR programs): Recall the POLAR program from Example 7 on page 112:

```
d1[ 0.6 soccer
    0.8 s1[ 0.3 music ]
    0.7 *a1]
a1[ 0.5 football ]
document(d1)
annotation(a1)
```

This example contains many important elements of POLAR programs like propositions (terms and classifications), documents, content annotations, and subparts. The corresponding FVPD translation is

```
1  0.6 term(soccer,d1).
2  0.8 acc_subpart(d1,s1).
3  0.7 acc_canno(d1,a1).
```

```
4   0.3 term(music,s1).
5   0.5 term(football,a1).
6       instance_of(d1,document,db).
7       instance_of(a1,annotation,db).
8       subpart(d1).
9       subpart(a1).
```

Lines 1 to 3 translate the propositions of the d1 context. The binary predicate `term` represents term propositions – in this example, line 1 means that "soccer" is a term of d1 with 0.6 probability. `acc_subpart` models probabilistic access to subparts as subcontexts, so line 2 means that d1 accesses s1 with a probability of 0.8. `acc_subpart` encodes the probabilistic access of subcontexts as well as the `_subpart` special proposition defined in Def. 6 on page 97. In a similar way, line 3 deals with a1 as subcontext of d1 – a1 is a content annotation of d1 and 0.7 is the probability that a1 is accessed from d1. Lines 4 and 5 represent "`s1[0.3 music]`" and "`a1[0.5 football]`", respectively. The last four lines express the db-context. The predicate `instance_of` is used for classifications. Line 6 says that d1 is an instance of the class "document" in the global database context db. Analogously, line 7 means that a1 is known as an annotation in context db. Lines 8 and 9 internally categorise d1 and a1 as subparts. The `subpart` relation is later used to distinguish between queries looking for fragments and those looking for documents or annotations. Each element in this relation has the probability 1 (in contrast to the `acc_subpart` relation).

Let us have a look at another example, this time taken from Section 5.2.1.2:

```
d1[ *a1 *a2 ]
a1[ moon_made_of_cheese ]
a2[ !moon_made_of_cheese ]
```

The translation into FVPD yields

```
1   acc_canno(d1,a1).
2   acc_canno(d1,a2).
3   term(moon_made_of_cheese,a1).
4   !term(moon_made_of_cheese,a2).
5   subpart(d1).
6   subpart(a1).
7   subpart(a2).
```

The translation goes analogously to the previous one. Line 4 is interesting because it contains a negation. Negated terms in POLAR are transformed into a negated `term` predicate in FVPD.

Another example from the previous chapter is

```
schalke
footballclub(schalke)
0.7/0.3 peter.supports(schalke)
```

We find all 3 kinds of propositions here. The translation outputs the following FVPD program:

```
1   term(schalke,db).
2   instance_of(schalke,footballclub,db).
3   0.7/0.3 attribute(supports,peter,schalke,db).
```

Line 3 shows how an attribute within a context (in this case the db-context) is expressed with the `attribute` predicate, which takes the attribute name, the object ID, a constant as attribute value and the context in which the proposition is made. Since the FVPD syntax also allows for weight lists, we can just copy the POLAR weight list here.

The next example deals with fragments:

```
d1[ 0.5 information
    0.8 f1|| 0.7 information  0.7 -*a1|| ]
a1[]
```

```
1  0.5 term(information,d1).
2  0.7 term(information,f1).
3  0.8 acc_fragment(d1,f1).
4  0.7 acc_negcanno(f1,a1).
5  0.7 acc_negcanno(d1,a1).
6      subpart(d1).
7      subpart(a1).
8      fragment(f1).
```

The first two lines are the propositions made in the d1 and f1 context, respectively. Line 3 denotes the fragment access from d1 to f1. The following two lines are interesting, since they realise the access of the negative content annotation a1 from f1 as well as from d1 due to fragment permeability. The last three lines internally categorise d1 and a1 as subparts and f1 as fragment.

6.2.1.2 Translation Rules

From the examples above we now come to the formulation of translation rules for basic POLAR knowledge modelling. We begin bottom-up with facts, factlists and propositions.

trans(factlist,c)	:=	*trans*(fact,c) *trans*(factlist',c)
		if factlist = fact factlist'
trans(fact,c)	:=	*trans*(proposition,c)
		if fact = proposition
	:=	!*trans*(proposition,c)
		if fact = !proposition
	:=	weightlist *trans*(proposition,c)
		if fact = weightlist proposition
trans(proposition,c)	:=	`term(termname,c).`
		if proposition = `termname`
	:=	`instance_of(obj-id,classname,c)`
		if proposition = `classname(obj-id)`
	:=	`attribute(attr-name,obj-id,constant,c)`
		if proposition = `obj-id.attr-name(constant)`

A fact is a proposition, a negated proposition or a proposition with a weight list. Propositions can be terms, classifications or attributes. In FVPD, "`term(termname,c)`" means that `termname` is a term in the context c. "`instance_of(obj-id,classname,c)`" says

that in context c, the object with the ID `obj-id` is instance of the class `classname`. "`attribute(attr-name,obj-id,constant,c)`" means that in context c, the attribute `attr-name` of the object identified by `obj-id` has the value `constant`.

The translation of references and annorefs goes as follows:

$$
\begin{array}{rcl}
\textit{trans}(\text{annoref,c}) & := & \text{weight } \texttt{acc\_canno(c,ca)}. \\
 & & \text{if annoref} = \text{weight } \texttt{*ca} \\
 & := & \text{weight } \texttt{acc\_negcanno(c,ca)}. \\
 & & \text{if annoref} = \text{weight } \texttt{-*ca} \\
 & := & \text{weight } \texttt{acc\_manno(c,ma)}. \\
 & & \text{if annoref} = \text{weight } \texttt{@ma} \\
 & := & \text{weight } \texttt{acc\_negmanno(c,ma)}. \\
 & & \text{if annoref} = \text{weight } \texttt{-@ma} \\
\textit{trans}(\text{reference,c}) & := & \text{weight } \texttt{acc\_reference(c,r)} \\
 & & \text{if reference} = \text{weight } \texttt{=>r}
\end{array}
$$

The "`acc_canno`" predicate models probabilistic access of a subcontext from a supercontext and states that the subcontext is a content annotation. "`acc_canno(c,ca)`" thus means that the subcontext (in this case content annotation) ca is accessed from the supercontext c (with the given two-valued access probability as weight). Similarly for negative content annotations ("`acc_negcanno`"), and positive and negative meta annotations ("`acc_manno`","`acc_negmanno`"). For references, "`acc_reference(r,c)`" means that r is referenced by c. Besides access relations, all these predicates encode the special propositions introduced in Def. 6 and always carry the access probability as their weight.

We now turn to the translation of subparts, fragments and merged targets.

$$
\begin{array}{rcl}
\textit{trans}(\text{mergedtarget,c}) & := & \text{weight } \texttt{acc\_mtarget(c,m)}. \\
 & & \quad \textit{trans}(\text{factlist,m}) \\
 & & \text{if mergedtarget} = \text{weight } \texttt{m<} \text{ factlist } > \\
\textit{trans}(\text{subpart,c}) & := & \texttt{subpart(s)}. \\
 & & \quad \text{weight } \texttt{acc\_subpart(c,s)}. \\
 & & \quad \textit{trans}(\text{contextprogram,s}) \\
 & & \text{if subpart} = \text{weight } \texttt{s[} \text{ contextprogram } \texttt{]} \\
\textit{trans}(\text{fragment,c}) & := & \texttt{fragment(f)}. \\
 & & \quad \text{weight } \texttt{acc\_fragment(c,f)}. \\
 & & \quad \textit{trans}(\text{factlist,f}) \; \textit{trans}(\text{annoref,f}) \; \textit{trans}(\text{annoref,c}) \\
 & & \text{if fragment} = \text{weight } \texttt{f||} \text{ factlist annoref } \texttt{||} \\
 & := & \texttt{fragment(f)}. \\
 & & \quad \text{weight } \texttt{acc\_fragment(c,f)}. \\
 & & \quad \textit{trans}(\text{factlist,f}) \\
 & & \text{if fragment} = \text{weight } \texttt{f||} \text{ factlist } \texttt{||}
\end{array}
$$

Like annotations and references, subparts, fragments and merged targets are special kinds of subcontexts. To distinguish between the access of distinct subcontexts, we again define special access predicates. For merged targets, the predicate "`acc_mtarget`" is used. For subparts, "`acc_subpart`" is taken and fragments are marked as such by the "`acc_fragment`" predicate. Furthermore, subparts and fragments are internally categorised as such with the "`subpart`" and "`fragment`" predicate. If a fragment has an annoref, we need to apply fragment permeability, so the annoref also belongs to the same context as the fragment.

Based on the definitions so far, we define the translations of contexts, contextprograms and context clauses:

$$
\begin{array}{rcl}
\textit{trans}(\text{context,c}) & := & \texttt{subpart(d).} \\
 & & \quad \textit{trans}(\text{mergedtarget,d})\ \textit{trans}(\text{contextprogram,d}) \\
 & & \text{if context} = \texttt{d[mergedtarget contextprogram]} \\
 & := & \texttt{subpart(d).}\ \textit{trans}(\text{contextprogram,d}) \\
 & & \text{if context} = \texttt{d[contextprogram]} \\
\textit{trans}(\text{contextprogram,c}) & := & \textit{trans}(\text{contextclause,c})\ \textit{trans}(\text{contextprogram',c}) \\
 & & \text{if contextprogram} = \text{contextclause contextprogram'} \\
\textit{trans}(\text{contextclause,c}) & := & \textit{trans}(\text{fact,c}) \\
 & & \text{if contextclause} = \text{fact} \\
 & := & \textit{trans}(\text{subpart,c}) \\
 & & \text{if contextclause} = \text{subpart} \\
 & := & \textit{trans}(\text{annoref,c}) \\
 & & \text{if contextclause} = \text{annoref} \\
 & := & \textit{trans}(\text{reference,c}) \\
 & & \text{if contextclause} = \text{reference} \\
 & := & \textit{trans}(\text{fragment,c}) \\
 & & \text{if contextclause} = \text{fragment}
\end{array}
$$

A context clause can be a fact, subpart, annoref, reference or fragment. We already discussed the translation of these elements. A context program is composed of context clauses. A context is similar to a subcontext except that it is only accessed from the db-context and can contain a merged target as a subcontext.

6.2.1.3 Preamble: Closed World Assumptions and Disjointness Keys

To finalise the translation of the basic knowledge modelling, we need to specify the relations a closed world is assumed for, and which relations contain disjoint facts. This definition must precede any FVPD translation of POLAR programs and is as follows:

```
1  _CWA(acc_subpart)
2  _CWA(acc_mtarget)
3  _CWA(acc_fragment)
4  _CWA(acc_canno)
5  _CWA(acc_negcanno)
6  _CWA(acc_manno)
7  _CWA(acc_negmanno)
8  _CWA(acc_reference)

10 _dk(term,"$1,$2")
11 _dk(instance_of,"$1,$2,$3")
12 _dk(attribute,"$1,$2,$3,$4")
```

In the first 8 lines we assume a closed world for all access relations. This is consistent with the definition of accessibility relations in POLAR, which are two-valued. The last three lines define the disjointness of terms, categorisations and attributes. As shown in Example 14, these disjointness declarations are crucial to ensure that the four truth values of a proposition are disjoint.

6.2.2 Queries and Rules

Before we start the discussion of the translation of POLAR queries and rules into FVPD, we give some examples. Recall that variables in POLAR and also FVPD begin with capital letters; during processing of rules and queries, the FVPD engine instantiates these variables with actual constants.

Example 16 (Translation of POLAR queries and rules): Recall the POLAR programs in Example 15 on page 146. For the first program, a query

```
?- D[*A] & A[football] & document(D).
```

would yield all documents with a content annotation about "football" (the query would actually return document-annotation pairs). The corresponding FVPD query is

```
1  ?- subpart(D) & acc_canno(D,A) & subpart(A) &
2     term(football,A) & instance_of(D,document,db).
```

The FVPD subgoal "`subpart(D) & acc_canno(D,A)`" is the translation of "`D[*A]`" and yields instantiations of the variables `D` and `A` which fulfil the criteria that `D` accesses `A` and `A` is a content annotation of `D`. "`subpart(A) & term(football,A)`" is the translation of the "`A[football]`" POLAR subgoal and returns all instantiations of `A` (and their respective probabilities) which contain the term "football". The translation of "`document(D)`" is "`instance_of(D,document,db)`"; this returns all instantiations of `D` which are an instance of the class "document" in the db-context. We used "`subpart(D)`" and "`subpart(A)`" to prohibit that fragments or merged targets are returned here. After processing, the FVPD engine returns the document-annotation pairs which make all subgoals true; if we consider the first program in Example 15 on page 146, this is

```
0.35 (d1,a1)
```

with the probability calculated as $0.7 \cdot 0.5 \cdot 1 = 0.35$; each FVPD subgoal (and thus each POLAR subgoal) is assumed to be an independent event, so the probability of the conjunction of these subgoals is the product of the single probabilities of each subgoal. Now consider two rules which say that an object should be returned if it is about "soccer" or has an annotation about "football":

```
retrieve(D) :- D[soccer]
retrieve(D) :- D[*A] & A[football]
```

This is translated into the FVPD rules

```
1  instance_of(D,retrieve,db) :- subpart(D) & term(soccer,D)
2  instance_of(D,retrieve,db) :- subpart(D) & acc_canno(D,A) &
3                                subpart(A) & term(football,A)
```

Line 1 says that an instantiation of the variable `D` is an instance of the class "retrieve" in the db-context if it is a subpart and contains the term "soccer". Line 2 and 3 says that it is also an instance of the class "retrieve" if it accesses a content annotation `A` and `A` contains the term "football". The FVPD engine would create the new FVPD fact "`instance_of(d1,retrieve,db)`" and thus the new POLAR fact "`retrieve(d1)`" with probability 0.74 ($0.6 + 0.7 \cdot 0.5 - 0.6 \cdot 0.7 \cdot 0.5 = 0.74$). Another rule might be

```
D[football] :- D[soccer]
```

which says that each document about 'soccer' is also one about 'football' (consider the European 'football' and the American 'soccer'). The translation of this rule is

```
1  term(football,D) :- subpart(D) & term(soccer,D).
```

and the FVPD engine would create the new fact "`0.6 term(football,d1)`", which in POLAR would be "`d1[0.6 football]`".

If we look at the second POLAR program in Example 15, we could ask for objects which say that the moon is not made of cheese:

```
?- D[ !moon_made_of_cheese ]
```

This POLAR query would be translated into

```
?- subpart(D) & !term(moon_made_of_cheese,D)
```

and would yield a2. The following POLAR rules say that a document should be retrieved if it states that the moon is not made of cheese, and it should not be retrieved if it asserts that the moon is made of cheese:

```
relevant(D) :- D[!moon_made_of_cheese]
!relevant(D) :- D[moon_made_of_cheese]
```

The corresponding FVPD translation is:

```
1  instance_of(D,retrieve,db) :-
2                    subpart(D) & !term(moon_made_of_cheese,D)
3  !instance_of(D,retrieve,db) :-
4                    subpart(D) & term(moon_made_of_cheese,D)
```

The negation within a context in the subgoal of the first POLAR rule is translated into a negated subgoal in the FVPD rule (lines 1 and 2). The negated head in the second POLAR rule is turned into a negated head in the second FVPD rule. The FVPD engine generates new facts equivalent to "`relevant(a2)`" and "`!relevant(a1)`".

As another example, consider the third program in Ex. 15. We can ask for all clubs which Peter does not support:

```
?- !peter.supports(C)
```

which is translated into

```
1  ?- !attribute(supports,peter,C,db)
```

and returns "schalke" with a probability of 0.3. □

Finally, let us have a look at fragments about 'information':

```
?- F||information||
```

This is translated to

```
1  ?- fragment(F) & term(information,F).
```

We can see here the role of the internal predicate "`fragment`" which is used to return fragments, but no subparts or merged targets. If we consider the fourth program in Example. 15, also d1 would be returned besides f1 if we omitted the "`fragment(F)`" subgoal.

6.2.2.1 Translation of Queries

We start with a discussion of queries in a top-down fashion.

$$
\begin{array}{rcl}
\textit{trans}(\text{query,c}) & := & \texttt{?-}\ \textit{trans}(\text{subgoallist,c}) \\
 & & \text{if query} = \texttt{?-}\ \text{subgoallist} \\
\textit{trans}(\text{subgoallist,c}) & := & \textit{trans}(\text{subgoal,c})\ \texttt{\&}\ \textit{trans}(\text{subgoallist',c}) \\
 & & \text{if subgoallist} = \text{subgoal}\ \texttt{\&}\ \text{subgoallist'} \\
 & := & \textit{trans}(\text{subgoal,c}) \\
 & & \text{if subgoallist} = \text{subgoal} \\
\textit{trans}(\text{subgoal,c}) & := & \texttt{subpart(s)}\ \texttt{\&}\ \textit{trans}(\text{csubgoallist,s,1}) \\
 & & \text{if subgoal} = \texttt{//s[}\ \text{csubgoallist}\ \texttt{]} \\
 & := & \texttt{subpart(s)}\ \texttt{\&}\ \textit{trans}(\text{csubgoallist,s,0}) \\
 & & \text{if subgoal} = \texttt{s[}\ \text{csubgoallist}\ \texttt{]} \\
 & := & \texttt{fragment(f)}\ \texttt{\&}\ \textit{trans}(\text{fsubgoallist,f}) \\
 & & \text{if subgoal} = \texttt{f||}\ \text{fsubgoallist}\ \texttt{||} \\
 & := & \textit{trans}(\text{factgoal,c,0}) \\
 & & \text{if subgoal} = \text{factgoal}
\end{array}
$$

In both POLAR and FVPD, queries start with "?-", so we just copy this sequence. It follows a translation of the subgoal list, which is a conjunction of one or more single subgoals. A subgoal can be a subpart with a special context subgoal list (csubgoallist), a fragment with a fragment subgoal list, or a fact goal. We introduce context subgoal lists because within subcontexts, different subgoals are possible than out of subcontexts; the same holds for fragments and fragment subgoal lists. A subpart subgoal can underlie knowledge augmentation (which is going to be discussed in the next section). In this case, we set the knowledge augmentation flag (see below). The translation of context subgoal and fragment subgoal lists goes as follows:

$$
\begin{array}{rcl}
\textit{trans}(\text{csubgoallist,c,k}) & := & \textit{trans}(\text{csubgoal,c,k})\ \texttt{\&}\ \textit{trans}(\text{csubgoallist',c,k}) \\
 & & \text{if csubgoallist} = \text{csubgoal}\ \texttt{\&}\ \text{csubgoallist'} \\
 & := & \textit{trans}(\text{csubgoal,c,k}) \\
 & & \text{if csubgoallist} = \text{csubgoal} \\
\textit{trans}(\text{csubgoal,c,k}) & := & \textit{trans}(\text{annorefgoal,c}) \\
 & & \text{if csubgoal} = \text{annorefgoal} \\
 & := & \textit{trans}(\text{referencegoal,c}) \\
 & & \text{if csubgoal} = \text{reference} \\
 & := & \textit{trans}(\text{fragmentgoal,c}) \\
 & & \text{if csubgoal} = \text{fragmentgoal} \\
 & := & \textit{trans}(\text{factgoal,c,k}) \\
 & & \text{if csubgoal} = \text{factgoal} \\
\textit{trans}(\text{fsubgoallist,c}) & := & \textit{trans}(\text{fsubgoal,c})\ \texttt{\&}\ \textit{trans}(\text{fsubgoallist',c}) \\
 & & \text{if fsubgoallist} = \text{fsubgoal}\ \texttt{\&}\ \text{fsubgoallist'} \\
 & := & \textit{trans}(\text{fsubgoal,c}) \\
 & & \text{if fsubgoallist} = \text{fsubgoal} \\
\textit{trans}(\text{fsubgoal,c}) & := & \textit{trans}(\text{annorefgoal,c}) \\
 & & \text{if fsubgoal} = \text{annorefgoal} \\
 & := & \textit{trans}(\text{factgoal,c}) \\
 & & \text{if fsubgoal} = \text{factgoal}
\end{array}
$$

The translation function for context subgoals has 3 parameters. The third parameter is the flag that controls knowledge augmentation. If set to 0, no knowledge augmentation should be

performed; if 1, knowledge augmentation should be performed. We need this parameter later in Section 6.2.3. Each context subgoal list is a conjunction of context subgoals. A context subgoal can be an annoref goal, a reference goal, a fragment goal or a fact goal. A fragment subgoal can be an annoref goal or a fact goal.

$$
\begin{array}{rcl}
\mathit{trans}(\text{annorefgoal,c}) & := & \texttt{acc\_canno(c,object)} \\
 & & \text{if annorefgoal} = \texttt{*object} \\
 & := & \texttt{acc\_negcanno(c,object)} \\
 & & \text{if annorefgoal} = \texttt{-*object} \\
 & := & \texttt{acc\_manno(c,object)} \\
 & & \text{if annorefgoal} = \texttt{@object} \\
 & := & \texttt{acc\_negmanno(c,object)} \\
 & & \text{if annorefgoal} = \texttt{-@object} \\
\mathit{trans}(\text{referencegoal,c}) & := & \texttt{acc\_reference(c,object)} \\
 & & \text{if referencegoal} = \texttt{=>object} \\
\mathit{trans}(\text{fragmentgoal,c}) & := & \texttt{acc\_fragment(c,object)} \\
 & & \text{if fragmentgoal} = \texttt{||object}
\end{array}
$$

Annoref, reference and fragment goals are used for queries which unveil annotation, reference or fragment relations. They produce a ranking according to the given access probability. The translation of these goals is a conjunction of the access probability and the relation type.

Fact goals are translated as follows:

$$
\begin{array}{rcl}
\mathit{trans}(\text{factgoal,c,k}) & := & \mathit{trans}(\text{atom,c,k}) \\
 & & \text{if factgoal} = \text{atom} \\
 & := & \texttt{!}\mathit{trans}(\text{atom,c,k}) \\
 & & \text{if factgoal} = \texttt{!}\,\text{atom} \\
\mathit{trans}(\text{atom,c,k}) & := & \texttt{term(termname,c)} \\
 & & \text{if atom} = \texttt{termname and k} = 0 \\
 & := & \texttt{instance\_of(object,classname,c)} \\
 & & \text{if atom} = \texttt{classname(object) and k} = 0 \\
 & := & \texttt{attribute(attr-name,object,} \\
 & & \qquad\qquad \texttt{attr-value,c)} \\
 & & \text{if atom} = \texttt{object.attr-name(attr-value)} \\
 & & \text{and k} = 0
\end{array}
$$

The rules above describe the case that no knowledge augmentation is performed, so the parameter k remains 0 (the knowledge augmentation case is discussed in Section 6.2.3). Fact goals can be atoms or negated atoms. Depending on atoms being terms, classifications or attributes, we query the corresponding predicate (`term`, `instance_of` or `attribute`).

6.2.2.2 Translation of Rules

The translation instructions for POLAR rules are as follows:

$$
\begin{array}{rcl}
\mathit{trans}(\text{rule,c}) & := & \mathit{trans}(\text{goal,c})\ \texttt{:-}\ \mathit{trans}(\text{subgoallist,c}) \\
\mathit{trans}(\text{goal,c}) & := & \mathit{trans}(\text{head,c}) \\
 & & \text{if goal} = \text{head} \\
 & := & !\mathit{trans}(\text{head,c}) \\
 & & \text{if goal} = !\,\text{head} \\
 & := & \mathit{trans}(\text{head,s}) \\
 & & \text{if goal} = \texttt{s[}\text{head}\texttt{]} \\
 & := & !\mathit{trans}(\text{head,s}) \\
 & & \text{if goal} = \texttt{s[!}\text{head}\texttt{]} \\
\mathit{trans}(\text{head,c}) & := & \texttt{term(termname,c)} \\
 & & \text{if head} = \texttt{termname} \\
 & := & \texttt{instance\_of(variable,classname,c)} \\
 & & \text{if head} = \texttt{classname(variable)} \\
 & := & \texttt{attribute(attr-name,variable,} \\
 & & \qquad\qquad \texttt{attr-value,c)} \\
 & & \text{if head} = \texttt{variable.attr-name(attr-value)}
\end{array}
$$

A goal can be a head within or outside a context. Each head may be negated. Heads are translated into a corresponding FVPD term, instance_of or attribute goal. The translation of subgoal lists has been discussed above.

6.2.3 Knowledge Augmentation

Knowledge augmentation is applied in queries or rules. As discussed in the previous chapter, the idea of knowledge augmentation is to augment our knowledge about a context with the knowledge (i.e. the propositions) contained in its subcontexts. Knowledge augmentation in POLAR is realised by providing FVPD support rules which can either be executed during indexing or query time. We have to cope with the peculiarities of the different subcontext types as discussed in the previous chapter – for instance, merged targets need to be treated differently than subparts. We further have to handle cycles on the access structure. To realise all this, the basic idea is to define new intensional FVPD predicates `term_k`, `instance_of_k` and `attribute_k`, which take the same arguments as `term`, `instance_of` and `attribute`, respectively. The difference is that the new intensional predicates contain the propositions and their respective probabilities of the augmented context. By augmenting the context, new facts are introduced to the original context relations and their probabilities are altered according to the evidence we find in the augmented context. During query time and when defining rules, users can decide whether to take the original context relation (e.g. by using an expression like "`D[t]`" which translates to "`term(t,D)`" for terms) to reflect non-augmented contexts, or the augmented context relation (e.g. using "`//D[t]`" which is translated to "`term_k(t,D)`") to consider augmented contexts in their rules or queries.

In the following we describe the support rules needed to implement knowledge augmentation as discussed in the last chapter. We begin with a set of recursive rules and prove their correctness w.r.t. the knowledge augmentation routines in Section 5.2.3. The rules can be adapted in a way that only the logical document structure, the annotation context or both are considered. In order to handle cycles, we introduce a serialisation strategy which has the additional advantage of being more efficient in most cases.

6.2.3.1 FVPD Support Rules for Knowledge Augmentation

To incorporate knowledge augmentation in our FVPD-based POLAR implementation, we must propagate the truth values *true*, *false* and, if the actual application demands, *inconsistent* from subcontexts to their respective supercontext. We first define some support rules in FVPD which realise the propagation of the truth value *true* with respect to the context type and discuss the application of these rules. We then present rules for the propagation of *false* and briefly discuss *inconsistent*. Similar rules must be defined for categorisations and attributes. The whole rule set is listed in Appendix B.1.

Propagation of *true* The following rules propagate the truth value *true* for terms in contexts and subcontexts.

```
1  term_k(T,D) :- term(T,D).
2  term_k(T,D) :- term_k_logical(T,D).
3  term_k(T,D) :- term_k_anno(T,D).
4  term_k(T,D) :- term_k_reference(T,D).

6  term_k_logical(T,D) :- acc_subpart(S,D) & term_k2(T,S).
7  term_k_anno(T,D) :- acc_mtarget(D,S) & term(T,S).
8  term_k_anno(T,D) :- acc_fragment(D,S) & term(T,S).
9  term_k_anno(T,D) :- acc_canno(D,S)  & term_k2(T,S).
10 term_k_anno(T,D) :- acc_negcanno(D,S) & !term_k2(T,S).
11 term_k_reference(T,D) :- acc_reference(D,S) & term_k2(T,S).



14 term_k2(T,D) :- term(T,D).
15 term_k2(T,D) :- term_k2_logical(T,D).
16 term_k2(T,D) :- term_k2_anno(T,D).
17 term_k2(T,D) :- term_k2_reference(T,D).

19 term_k2_logical(T,D) :- acc_subpart(D,S) & term_k2(T,S).
20 term_k2_anno(T,D) :- acc_fragment(D,S) & term(T,S).
21 term_k2_anno(T,D) :- acc_canno(D,S)  & term_k2(T,S).
22 term_k2_anno(T,D) :- acc_negcanno(D,S) & !term_k2(T,S).
23 term_k2_reference(T,D) :- acc_reference(D,S) & term_k2(T,S).
```

The FVPD rules simulate the behaviour of the *trees* function in Def. 17 on page 121 and Algorithm 1 on page 126 (except for the case that the access structure contains cycles, see below). We therefore need to define two sets of rules: one which defines the predicate `term_k` and marks the begin of the iteration; this set is introduced in lines 1 – 11. Lines 14 to 23 show the second set of rules where the support predicate `term_k2` is presented. For this set, no rule for merged targets is defined any more; this is due to the fact that merged targets are only accessed if they belong to the supercontext and ignored when they belong to a subcontext of the context we are augmenting (see the discussion in Section 5.2.3.2).

The rule in line 1 of the FVPD listing above simply says that each term appearing in a context also appears in its augmented context. Line 2 means that terms should be propagated if they are accessible through the logical structure of the document. The subsequent line 3 models the propagation of terms from subcontexts which are related to annotation. Line 4

propagates terms coming from referenced objects to the augmented context. The propagation from logical subcomponents, annotations and referenced objects is again expressed by rules. The rule in line 6 says that a term is part of the augmented context w.r.t. the logical structure if it appears in the augmented context of a subpart of the object to augment. Similarly in lines 7 – 10 for augmented subcontexts which are related to annotation. These are: merged targets (l. 7), fragments (l. 8), content annotations (l. 9) and negative content annotations (l. 10). The rule for fragments adheres to the definition of valid augmented context expressions which says that no subcontext of a fragment should be considered (merged targets do not have any subcontexts per definition); we therefore use the (non-recursive) `term` relation as a subgoal, instead of the recursive `term_k2` relation. Line 10 shows the propagation of terms in case the subcontext is a negative content annotation. As discussed in Section 5.2.3.2, we have positive evidence for a proposition if there is negative evidence in a negative content annotation subcontext, and vice versa. So a term should be *true* in an augmented context if it is *false* in the negative content annotation. Therefore, "`!term_k(T,S)`" is used as a subgoal. Finally, line 11 realises the propagation of terms from augmented subcontexts established by referenced objects. The description of lines 14 to 23 goes analogously. As outlined in Section 5.2.3.2, there is deliberately no rule for meta annotations, because their content is not subject to knowledge augmentation.

The rules realise the propagation of terms through the definition of a new intensional predicate `term_k`. In the specification of the support rules, we distinguish between rules realising access through the logical structure, the annotation context or the inter-document context (through referenced objects). This way we are able to adapt our retrieval strategy on the fly[3], for example if we only want to consider the logical structure of documents (e.g., for classical XML or structured document retrieval) and not their annotations, or vice versa. This provides us with suitable means to support the special command controlling the augmentation behaviour introduced in Section 4.3.1.2. For instance, if "`_no_structure_propagation()`" is specified, we just delete the rule in line 2. "`_no_annotation_propagation()`" means ignoring line 3, while "`_no_reference_propagation()`" would keep line 4 from being evaluated.

Application of Support Rules The `term_k` relation defined by the support rules can be applied when knowledge augmentation is requested. A POLAR query

```
?- D[football]
```

without knowledge augmentation would be translated into the FVPD query

```
1  ?- subpart(D) & term(football,D)
```

whereas the POLAR query

```
?- //D[football]
```

which requests knowledge augmentation, would be translated into the FVPD query

```
1  ?- subpart(D) & term_k(football,D)
```

forcing the FVPD engine to evaluate the support rules above (this evaluation actually leads to the evaluation of the corresponding PD translation which is outlined in Example 14).

Since the evaluation of the support rules during query time and the definition of `term_k` as an intensional relation might lead to high response times, we can make use of a property

[3]In case we apply the support rules during query time, that is

of knowledge augmentation, namely that it is query-independent. This means that we can carry out knowledge augmentation during indexing time and store term_k as an extensional relation, so we have to apply the support rules only once. During query time we just have to link the term_k relation. If we can detect all objects affected by changes in our structured annotation hypertext (e.g., new annotations, new documents), it is sufficient to apply knowledge augmentation for these objects only, which allows for incremental indexing.

Propagation of *false* and *inconsistent* The rules above propagate the truth value *true* in subcontexts to the augmented context. We can apply similar rules for the truth value *false*:

```
1  !term_k(T,D) :- !term(T,D).
2  !term_k(T,D) :- !term_k_logical(T,D).
3  !term_k(T,D) :- !term_k_anno(T,D).
4  !term_k(T,D) :- !term_k_reference(T,D).

6  !term_k_logical(T,D) :- acc_subpart(S,D) & !term_k2(T,S).
7  !term_k_anno(T,D) :- acc_mtarget(D,S) & !term(T,S).
8  !term_k_anno(T,D) :- acc_fragment(D,S) & !term(T,S).
9  !term_k_anno(T,D) :- acc_canno(D,S)  & !term_k2(T,S).
10 !term_k_anno(T,D) :- acc_negcanno(D,S) & term_k2(T,S).
11 !term_k_reference(T,D) :- acc_reference(D,S) & !term_k2(T,S).



14 !term_k2(T,D) :- !term(T,D).
15 !term_k2(T,D) :- !term_k2_logical(T,D).
16 !term_k2(T,D) :- !term_k2_anno(T,D).
17 !term_k2(T,D) :- !term_k2_reference(T,D).

19 !term_k2_logical(T,D) :- acc_subpart(D,S) & !term_k2(T,S).
20 !term_k2_anno(T,D) :- acc_fragment(D,S) & !term(T,S).
21 !term_k2_anno(T,D) :- acc_canno(D,S)  & !term_k2(T,S).
22 !term_k2_anno(T,D) :- acc_negcanno(D,S) & term_k2(T,S).
23 !term_k2_reference(T,D) :- acc_reference(D,S) & !term_k2(T,S).
```

Inconsistent knowledge is not propagated by applying the rules so far. First of all we have to decide if inconsistent knowledge should really be propagated from subcontexts to supercontexts. This depends on the actual application. Second, the realisation of the propagation of *inconsistent* is not directly supported by FVPD. This is due to the fact that a FVPD subgoal like "`term_k2(T,S)`" is translated into the PD subgoal "`pos_term_k2(T,S) & !neg_term_k2(T,S)`", as we have seen in Section 6.1.2. If we want inconsistent knowledge to be propagated, we must use a PD translation like "`pos_term_k2(T,S) & neg_term_k2(T,S)`" instead, which contradicts to the translation of FVPD into PD. So we either translate our POLAR augmentation rules directly into PD to support subgoals propagating inconsistent knowledge, or, alternatively, use the transitive closure over the acc relations, as it is proposed in (Rölleke, 1998, p. 135).

Correctness of Rules We have to show that the rules above and in Appendix B.1, respectively, perform knowledge augmentation in the sense that they calculate the same probability as

World	Case/Event	Probability
w_1^*	t1 *true* in d, no further access	$0.6 \cdot 0.7 \cdot 0.2 = 0.084$
w_2^*	t1 *true* in d, acc. a1, t1 *unknown* in a1, no further access	$0.6 \cdot 0.3 \cdot 0.6 \cdot 0.2 = 0.0216$
w_3^*	t1 *true* in d, acc. a1, t1 *true* in a1, no further access	$0.6 \cdot 0.3 \cdot 0.4 \cdot 0.2 = 0.0144$
w_4^*	t1 *true* in d, acc. a1, t1 *unknown* in a1, acc. a2, t1 *unkn.* in a2	$0.6 \cdot 0.3 \cdot 0.6 \cdot 0.8 \cdot 0.8 = 0.06912$
w_5^*	t1 *true* in d, acc. a1, t1 *unknown* in a1, acc. a2, t1 *true* in a2	$0.6 \cdot 0.3 \cdot 0.6 \cdot 0.8 \cdot 0.2 = 0.01728$
w_6^*	t1 *true* in d, acc. a1, t1 *true* in a1, acc. a2, t1 *unknown* in a2	$0.6 \cdot 0.3 \cdot 0.4 \cdot 0.8 \cdot 0.8 = 0.04608$
w_7^*	t1 *true* in d, acc. a1, t1 *true* in a1, acc. a2, t1 *true* in a2	$0.6 \cdot 0.3 \cdot 0.4 \cdot 0.8 \cdot 0.2 = 0.01152$
w_8^*	t1 *true* in d, acc. a2, t1 *unknown* in a2, no further access	$0.6 \cdot 0.7 \cdot 0.8 \cdot 0.8 = 0.2688$
w_9^*	t1 *true* in d, acc. a2, t1 *true* in a2, no further access	$0.6 \cdot 0.7 \cdot 0.8 \cdot 0.2 = 0.0672$
w_{10}^*	t1 *unknown* in d, no further access	$0.4 \cdot 0.7 \cdot 0.2 = 0.056$
w_{11}^*	t1 *unknown* in d, acc. a1, t1 *unknown* in a1, no further access	$0.4 \cdot 0.3 \cdot 0.6 \cdot 0.2 = 0.144$
w_{12}^*	t1 *unknown* in d, acc. a1, t1 *true* in a1, no further access	$0.4 \cdot 0.3 \cdot 0.4 \cdot 0.2 = 0.0096$
w_{13}^*	t1 *unknown* in d, acc. a1, t1 *unkn.* in a1, acc. a2, t1 *unkn.* in a2	$0.4 \cdot 0.3 \cdot 0.6 \cdot 0.8 \cdot 0.8 = 0.04608$
w_{14}^*	t1 *unknown* in d, acc. a1, t1 *unkn.* in a1, acc. a2, t1 *true* in a2	$0.4 \cdot 0.3 \cdot 0.6 \cdot 0.8 \cdot 0.2 = 0.01152$
w_{15}^*	t1 *unknown* in d, acc. a1, t1 *true* in a1, acc. a2, t1 *unkn.* in a2	$0.4 \cdot 0.3 \cdot 0.4 \cdot 0.8 \cdot 0.8 = 0.03072$
w_{16}^*	t1 *unknown* in d, acc. a1, t1 *true* in a1, acc. a2, t1 *true* in a2	$0.4 \cdot 0.3 \cdot 0.4 \cdot 0.8 \cdot 0.2 = 0.00768$
w_{17}^*	t1 *unknown* in d, acc. a2, t1 *unknown* in a2, no further access	$0.4 \cdot 0.7 \cdot 0.8 \cdot 0.8 = 0.1792$
w_{18}^*	t1 *unknown* in d, acc. a2, t1 *true* in a2, no further access	$0.4 \cdot 0.7 \cdot 0.8 \cdot 0.2 = 0.0448$

Table 6.1: Worlds based on *trees* output for our example

described in Section 5.2.3.2, if the access structure does not contain cycles (the case that the access structure contains cycles is discussed below).

Proposition 3 (Correctness of knowledge augmentation rules): *The rules for knowledge augmentation are correct if the access structure does not contain cycles.*

Proof. We prove the correctness of our rules using induction. Consider an augmented context $d(s_1, \ldots, s_n)$ such that $s_1, \ldots, s_n$ are direct subcontexts of d (which might be augmented contexts as well). We prove the correctness of our augmentation rules for *true* propositions (in this case terms) φ; the same proof can be applied for *false* propositions. We start with an augmented context containing no subcontext ($n = 0$) and induce over the number n of considered subcontexts. The idea is to show that the probability of the FVPD event expression which is created when evaluating n subcontexts always calculates the correct result for $d(s_1, \ldots, s_n)$. To illustrate the proof, we use the following simple example program:

```
d1 [ 0.6 t1   0.3 *a1   0.8 *a2 ]
a1 [ 0.4 t1 ]
a2 [ 0.2 t1 ]
```

Table 3 shows the 18 worlds created by *trees* for the example, the cases they represent and their probabilities.

$\boldsymbol{n = 0}$: No subcontext of d is considered, so the augmented context is $d()$. In this case, φ is only *true* in $d()$ if it is *true* in d, so we take the corresponding probability from d (which is the sum of the probabilities of worlds w_1^* – w_9^* in the example). This is exactly what the rule in line 1 does; the corresponding event expression is `term(`φ`,d)`. Therefore, $P($`term_k(`φ`,d)`$) = P($`term(`φ`,d)`$)$, which is the probability that φ is *true* in d. The augmentation is correct if no subcontext is considered.

$\boldsymbol{n = 1}$: Now we present the case that we consider one subcontext s, so we create the augmented context $d(s)$. According to the discussion in Section 5.2.3.2, the probability that φ

is *true* in $d(s)$ is the sum of the probabilities of the worlds w^* represented by the corresponding G-agent-world-polarity-trees in which d and s are involved, so that each of these worlds stands for a case which makes φ *true* in the augmented context. If φ is the only proposition appearing in d and s, these worlds are:

World	Case/Event
w_1^{**}	φ is *true* in d AND NOT d accesses s
w_2^{**}	φ is *true* in d AND d accesses s AND φ is *unknown* in s
w_3^{**}	φ is *true* in d AND d accesses s AND φ is *true* in s
w_4^{**}	φ is *unknown* in d AND d accesses s AND φ is *true* in s

In case s is a negative content annotation, we set "φ *is false in* s" instead of "φ *is true in* s". Note that s might be an augmented context itself. w_1^{**} combines all worlds where φ is *true* in d and d does not access s. In our example, if s = a1, these are the worlds w_1^*, w_8^* and w_9^*. $P(w_1^{**})$ is the sum of the probabilities of the worlds w_1^{**} is composed of. w_2^{**} is composed of w_2^*, w_4^* and w_5^*, w_3^{**} contains w_3^*, w_6^* and w_7^*, and w_4^{**} is composed of w_{12}^*, w_{15}^* and w_{16}^*. Each of the four worlds thus describes a basic or composed event; all these events are disjoint, so for two worlds w_i^{**} and w_j^{**} $P(w_i^{**} \vee w_j^{**}) = P(w_i^{**}) + P(_j^{**})$ if $i \neq j$. From the table above, we can derive the probability of the event "φ *is true in* d", which we denote by Φ_d:

$$P(\Phi_d) = P(w_1^{**} \vee w_2^{**} \vee w_3^{**}) = P(w_1^{**}) + P(w_2^{**}) + P(w_3^{**}).$$

Note that in our example, Φ_d consists of the worlds w_1^* – w_9^*. Another event is "d *accesses* s *AND* φ *is true in* s", which we call Φ_s. Then

$$P(\Phi_s) = P(w_3^{**} \vee w_4^{**}) = P(w_3^{**}) + P(w_4^{**}).$$

Φ_s comprises the worlds $w_3^*, w_6^*, w_7^*, w_{12}^*, w_{15}^*$ and w_{16}^* in our example. Furthermore, $\Phi_d \vee \Phi_s$ denotes the event that "φ *is true in* d *OR* d *accesses* s *AND* φ *is true in* s". Then

$$\begin{aligned} P(\Phi_d \vee \Phi_s) &= P(w_1^{**} \vee w_2^{**} \vee w_3^{**} \vee w_3^{**} \vee w_4^{**}) \\ &= P(w_1^{**} \vee w_2^{**} \vee w_3^{**} \vee w_4^{**}) \\ &= P(w_1^{**}) + P(w_2^{**}) + P(w_3^{**}) + P(w_4^{**}) \\ &= P(\Phi_d) + P(\Phi_s) - P(w_3^{**}). \end{aligned}$$

Note that w_3^{**} denotes the events/worlds common in Φ_d and Φ_s (w_3^*, w_6^* and w_7^* in the example).

We now turn to the evaluation of the FVPD rules for our case. Considering one subcontext s means that during the evaluation of `term_k`, the variable `S` is substituted with s, but has not been substituted yet with any other possible subcontext. The rule in line 1 and one (and only one) of the rules in the lines 7 to 11 contribute to the calculation of the probability of φ to be *true* in $d(s)$. The corresponding FVPD event expression is[4] `term(`φ`,d)` $\vee$ `acc(d,s)` $\wedge$ `term(`φ`,s)`. Then

$$\begin{aligned} &P(\texttt{term(}\varphi\texttt{,d)} \vee \texttt{acc(d,s)} \wedge \texttt{term(}\varphi\texttt{,s)}) = \\ &\quad P(\texttt{term(}\varphi\texttt{,d)}) + P(\texttt{acc(d,s)} \wedge \texttt{term(}\varphi\texttt{,s)}) - \\ &\quad P(\texttt{term(}\varphi\texttt{,d)} \wedge \texttt{acc(d,s)} \wedge \texttt{term(}\varphi\texttt{,s)}). \end{aligned}$$

[4] We omit the relation type in the event expression and just use `acc` instead of, e.g., `acc_canno`

`term(φ,d)` describes the event that φ appears in d, so $P(\texttt{term}(\varphi,\texttt{d})) = P(\Phi_d)$. Furthermore, $P(\texttt{acc(d,s)} \wedge \texttt{term}(\varphi,\texttt{s})) = P(\Phi_s)$, $P(\texttt{term}(\varphi,\texttt{d}) \wedge \texttt{acc(d,s)} \wedge \texttt{term}(\varphi,\texttt{s})) = P(w_3^*)$[5] and thus

$$P(\texttt{term}(\varphi,\texttt{d}) \vee \texttt{acc(d,s)} \wedge \texttt{term}(\varphi,\texttt{s})) = P(\Phi_d \vee \Phi_s).$$

The rules for knowledge augmentation are correct for a context with one subcontext.

$\boldsymbol{n \rightsquigarrow n+1}$: Now assume the rules for knowledge augmentation are correct for n subcontexts, $d(s_1,\ldots,s_n)$. We infer the correctness for $n+1$ subcontexts by adding a new subcontext s_{n+1}. The following events and worlds denote all events which let φ come *true*:

World	Case/Event
w_1^{**}	φ is *true* in $d(s_1,\ldots,s_n)$ AND NOT d accesses s_{n+1}
w_2^{**}	φ is *true* in $d(s_1,\ldots,s_n)$ AND d accesses s_{n+1} AND φ is *unknown* in s_{n+1}
w_3^{**}	φ is *true* in $d(s_1,\ldots,s_n)$ AND d accesses s_{n+1} AND φ is *true* in s_{n+1}
w_4^{**}	φ is *unknown* in $d(s_1,\ldots,s_n)$ AND d accesses s_{n+1} AND φ is *true* in s_{n+1}

Again, the worlds may be composed of other worlds. For example, w_1^{**} combines all worlds created by *trees* where φ is *true* with respect to d and the contexts $s_1,\ldots,s_n$, and d does not access s_{n+1}. As above, we get

$$\begin{aligned} P(\Phi_{d(s_1,\ldots,s_n)} \vee \Phi_{s_{n+1}}) &= P(w_1^{**} \vee w_2^{**} \vee w_3^{**} \vee w_3^{**} \vee w_4^{**}) \\ &= P(\Phi_{d(s_1,\ldots,s_n)}) + P(\Phi_{s_{n+1}}) - P(w_3^{**}) \end{aligned}$$

On the FVPD side, let `term_k'(φ,d)` be the event expression created for the calculation of $d(s_1,\ldots,s_n)$. The partial evaluation of the knowledge augmentation rules for d and φ would calculate an intermediate result $P(\texttt{term\_k'}(\varphi,\texttt{d}))$ based on the subcontexts $s_1,\ldots,s_n$; i.e., the variable `S` in the augmentation rules would have been substituted with $s_1,\ldots,s_n$ so far and is now substituted with s_{n+1}. $P(\texttt{term\_k'}(\varphi,\texttt{d}))$ denotes the probability that φ is *true* in the augmented context with respect to $s_1,\ldots,s_n$, which is the same as the probability $P(\Phi_{d(s_1,\ldots,s_n)})$. Let `term_k(φ,d)` be the event expression created when evaluating $d(s_1,\ldots,s_{n+1})$. Then $P(\texttt{term\_k}(\varphi,\texttt{d}))$ is the probability that φ is *true* in the augmented context with respect to $d(s_1,\ldots,s_{n+1})$. We get

$$\begin{aligned} &P(\texttt{term\_k}(\varphi,\texttt{d})) = P(\texttt{term\_k'}(\varphi,\texttt{d}) \vee \texttt{acc(d,s)} \wedge \texttt{term}(\varphi,\texttt{s})) = \\ &\quad P(\texttt{term\_k'}(\varphi,\texttt{d})) + P(\texttt{acc(d,s)} \wedge \texttt{term}(\varphi,\texttt{s})) - \\ &\qquad P(\texttt{term\_k'}(\varphi,\texttt{d}) \wedge \texttt{acc(d,s)} \wedge \texttt{term}(\varphi,\texttt{s})) = \\ &\quad P(\Phi_{d(s_1,\ldots,s_n)}) + P(\Phi_{s_{n+1}}) - P(w_3^{**}) = P(\Phi_{d(s_1,\ldots,s_n)} \vee \Phi_{s_{n+1}}). \end{aligned}$$

$P(\texttt{term\_k}(\varphi,\texttt{d})) = P(\Phi_{d(s_1,\ldots,s_n)} \vee \Phi_{s_{n+1}})$, so our knowledge augmentation rules are correct for positive propositions. The correctness for negative propositions as well as classifications and attributes can be shown analogously. ∎

[5] We use `term_k2(φ,s)` instead of `term(φ,s)` if s is an augmented context as well (which happens when processing subcontexts which are not merged targets or fragments). In that case, we assume that `term_k2(φ,s)` is fully evaluated and its probability calculated when it is applied for further computation. `term_k2(φ,s)` is not a basic event expression in its classic sense, but a placeholder for the event expression that led to the calculation of its probability.

Serialisation As outlined in Section 5.2.3.2, there can be cycles in our access structure. While the support rules above work well in cases we can ensure there are no cycles (for example if there are no reference links), we have to cope with the situation that cycles occur. Recall our POLAR example from the last chapter, enriched with a term in d1:

```
d1[ *a1 t1 ]
a1[ 0.8 *a2 ]
a2[ =>d1 ]
```

This would be translated to

```
1      term(t1,d1).
2      acc_canno(d1,a1).
3  0.8 acc_canno(a1,a2).
4      acc_reference(a2,d1).
5      subpart(d1)
6      subpart(a1)
7      subpart(a2)
```

When applied, our knowledge augmentation rules would potentially run into an infinite loop. For instance, if we want to calculate the augmented context of d1, the evaluation of `term_k(d1,t1)` would invoke `term_k2(a1,t1)` which in turn would result in the evaluation of `term_k2(a2,t1)`, `term_k2(d1,t1)` and again `term_k2(a1,t1)`. In a non-probabilistic case, the cycle might be detected when the underlying FVPD engine is applying fixpoint iteration (Ceri et al., 1990, section 7.2), but we cannot rely on it, especially in the probabilistic case, in which fixpoint iteration would not work. Furthermore, since we deal with negated subgoals, the resulting FVPD program might not be modularly stratified if the access structure (that is, all `acc_*` relations) is cyclic (Ross, 1990). One of our options is to *serialise* the knowledge augmentation rules and run special serialised rules for each context we like to augment.

Serialisation works as follows. We first have to analyse the structure established by all subcontexts accessed by the context to augment (d1 here). The result of such an analysis is an augmented context expression which does not contain any loops within a path (similar to those created by **createAugmContextExpr** (Alg. 1) based on the possible world semantics). We outline such an analysis algorithm below. For our example above, the cycle-free augmented context expression is d1(a1(a2)). The augmented context expression encodes all paths which are visited during knowledge augmentation. It controls the serialisation of the support rules in a similar way as it controlled the *trees* function in Def. 17. In our example expression, we find one path d1 – a1 – a2. We now create a set of rules for each path. Each path then contributes to the final augmentation result. For our example, we gain:

```
1  # rule for a2 (1st iteration)
2  term_k_it1(T,a2) :- term(T,a2).
3
4  # rules for a1(a2) (2nd iteration)
5  term_k_it2(T,a1) :- term(T,a1).
6  term_k_anno_it2(T,a1) :- acc_canno(a1,a2) & term_k_it1(T,a2).
7  term_k_it2(T,a1) :- term_k_anno_it2(T,a1).
```

```
9  # rules for d1(a1(a2)) (3rd iteration)
10 term_k(T,d1) :- term(T,d1).
11 term_k_anno(T,d1) :- acc_canno(d1,a1) & term_k_it2(T,a1).
12 term_k(T,d1) :- term_k_anno(T,d1).
```

The sequence of rules is crucial. To derive the rules from the augmented context expression d1(a1(a2)), we begin with the last element on each path. In our example above, this is a2. Since from a2 we do not access any more subcontext, its augmented context contains only propositions found in a2 itself. The only rule we create for this object is seen in line 2. It creates a new intensional relation `term_k_it1` which contains all terms found in a2 (note that we pre-initialise the second argument in `term_k_it1` and `term` with the actual object ID, in this case `a2`, instead of using a variable). Each such intensional relation gets a unique id, in this case `it1`. This is necessary since the same subcontext might be handled differently in different paths, so we need to distinguish them. The next element to process is a1. The augmented context of a1, a1(a2), contains propositions from a1 (line 5) and a2 (lines 6 and 7)[6]. The rules in lines 10 – 12 are created in the next iteration step when handling d1; they compute the desired augmented context of d1, d1(a1(a2)). When creating the serialised rules, we must also take care of the constraints for the several subcontext types discussed in the last chapter – no meta annotations and only merged targets of the context to augment should be considered and from fragments, no subcontext is accessed.

When the rules in line 10 and 12 are executed, all data relevant to these rules has been computed before due to the serialised execution of the above rule set. The serialised execution starts with the rule in line 2 and calculates all intensional facts needed by the rule in line 7. Then, the rules 5 – 7 compute the intensional facts for the augmented context a1(a2). In the next step, the rule in line 10 adds all terms appearing in d1 to the augmented context d1(a1(a2)). Rules 11 and 12 add the terms of the augmented context a1(a2) to the augmented context d1(a1(a2)). The execution would stop then. The newly created facts can now be stored externally and be linked during query execution[7].

As mentioned above, an external algorithm must analyse the structure established by all subcontexts accessed by the context to augment. Similar to Alg. 1 on page 126 (but this time based on a POLAR program and not possible worlds), such an algorithm creates a pruned version of the augmented context expression by traversing the subcontexts of the context to augment. The algorithm traverses all paths of access and stops as soon as there is no other subcontext any more or it re-enters an object already on the path. For example, if we want to augment d1 in the above example, we find one infinite path d1 – a1 – a2 – d1 – ... which must be pruned. When the algorithm reaches d1 the second time, it should stop and return d1(a1(a2)). If an annotation can only have one annotation target and there are no references, the algorithm can be based on a depth first traversal (Güting, 1992, p. 152) of the access structure. Such a traversal creates a spanning tree, with the context to augment as root of the tree, and thus contains all subcontexts, directly or indirectly accessible by the context to augment, once. But there might be cases that a subcontext is visited more than once, for example when an annotation is a subcontext two or more times (see the discussion about access from multiple contexts on page 5.2.3.2) or a subcontext is referenced several times or referenced object and

[6] Line 7 is of course redundant and only shown for the sake of readability. An actual implementation might merge the rules in lines 6 and 7 to
`term_k_it2(T,a1) :- acc_canno(a1,a2) & term_k_it1(T,a2).`

[7] The HySpirit distribution, which also contains an FVPD engine, offers means to dynamically link sets of facts and rules. It also offers means for serialised execution.

Parameters: POLAR program P, context to augment c, path p
Return: Augmented context expression $expr_c$
$expr_c := c$
$SUB :=$ getSubcontexts(P,c) {SUB contains all considerable direct subcontexts of c in P}
if $SUB \neq \emptyset$ **then** {There are subcontexts to access}
$p :=$ addToPath(p,c) {Add c to current path}
for all $s \in SUB$ **do**
if !contained(s,p) **then** {s not in path yet}
$expr_c :=$ *concat*($expr_c$, createAugmContextExpr2(P,s,p)) {add augmented subcontext expression to context expression}
end if
end for
end if

Algorithm 2: Algorithm **createAugmContextExpr2** to create pruned augmented context expressions

annotation at the same time. Consider another POLAR example, where d2 is referenced by both a1 and a2:

```
d1[ *a1 ]
a1[ *a2 =>d2 ]
a2[ =>d1 =>d2 ]
d2[]
```

We find 3 paths here:
d1 – a1 – a2 – d1 – ...
d1 – a1 – a2 – d2
d1 – a1 – d2

The algorithm should prune the first path[8] to d1 – a1 – a2. Since this path is contained in the second path above, the algorithm should remove this path completely. We see that d2 is contained in paths 2 and 3. A depth-first traversal as described in (Güting, 1992, p. 152) would erase one of the occurrences of d2, but this is not what we want. The algorithm **createAugmContextExpr2**, which is shown in Alg. 2, creates the augmented context expression as desired. The algorithm is firstly invoked with the context we want to augment and an empty path p. In contrast to a regular depth-first traversal, the algorithm only checks if a subcontext appears in the current path, not if it has been visited before in the whole process. A subcontext might thus appear several times in the augmented context expression. The algorithm returns the augmented context expression which is then the basis for the creation of the serialised support rules as described above.

Besides being able to handle cycles in our access structure, the serialisation offers another advantage: due to the creation of different intensional relations per iteration step and object found on a path, combined with the pre-instantiation of variables with the object ID, these relations contain a relatively small amount of tuples. This means that a faster processing of these rules is possible. For instance, the rule in line 6 contains a Join which we can suppose to be processed very quickly since only few tuples are involved in the Join operation. So it makes sense to apply serialisation even if we are sure that no cycles can occur.

[8] Actually, this path represents an unlimited number of path.

To create the serialised support rules, the above procedure has to be applied repeatedly for every context (document and annotation) in the POLAR knowledge base. For each such context a set of rules is created which handles its augmentation. If we have to deal with a huge amount of contexts, it is preferable to build and apply the serialised rules during indexing. For each context, the corresponding serialised rules could be created, then applied by an engine, and the resulting `term_k` relation could be stored externally. The created rules are not needed any more for another context, so we do not have to store them.

Radius-1 Augmentation Another option for knowledge augmentation is *radius-1 augmentation*. Here, we do not consider all subcontexts which are reachable by the context to augment, but only propagate knowledge coming from direct subcontexts. One advantage is that this kind of augmentation is more efficient, since only direct subcontexts have to be visited and processed. Another advantage is that there cannot occur any cycle. The disadvantage is that we neglect every indirect subcontext. This might have a negative effect on the retrieval quality. Especially in the case when we want to augment an object with its logical document structure, it may be dangerous to ignore parts of the whole document which are not accessed directly. A hybrid strategy (performing radius-1 augmentation with annotations and referenced objects, and full augmentation with the logical document structure) may be applied here.

Radius-1 augmentation can be performed either with serialised rules or with the non-serialised ones presented on page 156. In both cases, the augmentation rules only have to consider the knowledge in the direct subcontext. For example, for the non-serialised rules, we would not need the relation `term_k2` any more. The rules for term augmentation based on the logical structure, the annotation and the referenced objects would be:

```
1 term_k_logical(T,D) :- acc_subpart(S,D) & term(T,S).
2 term_k_anno(T,D) :- acc_mtarget(D,S) & term(T,S).
3 term_k_anno(T,D) :- acc_fragment(D,S) & term(T,S).
4 term_k_anno(T,D) :- acc_canno(D,S)  & term(T,S).
5 term_k_anno(T,D) :- acc_negcanno(D,S) & !term(T,S).
6 term_k_reference(T,D) :- acc_reference(D,S) & term(T,S).
```

6.2.3.2 Translation Rules for Knowledge Augmentation

After having discussed knowledge augmentation and the `term_k` relation, we need to integrate this into the *trans* function. A directive to perform knowledge augmentation can be given in rule and query subgoals. To consider this, we have to extend the translation of atoms from Section 6.2.2.1 as follows:

$$
\begin{aligned}
\mathit{trans}(\text{atom,c,k}) \;:=\; & \texttt{term\_k(termname,c)} \\
& \text{if atom} = \texttt{termname} \text{ and } k = 1 \\
:=\; & \texttt{instance\_of\_k(object,classname,c)} \\
& \text{if atom} = \texttt{classname(object)} \text{ and } k = 1 \\
:=\; & \texttt{attribute\_k(attr-name,object,} \\
& \qquad \texttt{attr-value,c)} \\
& \text{if atom} = \texttt{object.attr-name(attr-value)} \\
& \text{and } k = 1
\end{aligned}
$$

We use the predicates `term_k`, `instance_of_k` and `attribute_k` instead of `term`, `instance_of` and `attribute` if knowledge augmentation is activated.

6.2.4 Retrieval Function

So far, we introduced the translation of basic knowledge modelling, knowledge augmentation, queries and rules from POLAR into FVPD. While with these mechanisms we are able to pose many different kinds of queries to the underlying knowledge base, we still need to define retrieval functions based on uncertain inference as discussed in Section 5.3. The POLAR expression

```
D->q
```

should, when evaluated, calculate $P(d \rightarrow q)$ for each instantiation d of `D`. In principle, we can apply the same translation as described in (Rölleke, 1998, section 6.3) for POOL, except that in POLAR we have distinguished expressions for the application of knowledge augmentation – the expression "`//D->q`" instructs POLAR to take the augmented context of d instead of d alone.

Before we explain possible translations of `D->q` into FVPD-based retrieval functions, we start with an example. As mentioned in Section 5.3.2.1, queries are modelled as contexts as well. Consider the following POLAR program (with q1 as the query):

```
d1[ 0.6 ir  0.8 db ]
q1[ 0.6 ir  0.9 db ]

?- D->q1
```

This is translated into the following FVPD program:

```
1  0.6 term(ir,d1).
2  0.8 term(db,d1).
3  0.6 term(ir,q1).
4  0.9 term(db,q1).

6  ?- implies(D,q1).
```

The first two lines are the representation of d1 in FVPD. The next two lines represent q1. The last line shows the translation of "`?- D->q1`". This simply contains a new predicate `implies`. In this predicate lies the magic of our retrieval approach. Similar to the definition of `term_k` as an intensional predicate with rules, we can define support rules for `implies` which are set up and executed during query time, and thus easily integrate new retrieval functions into the framework. This way we can customise the interpretation of probabilistic implication w.r.t. the given application – we just need to define FVPD rules for `implies` which must be linked to the actual POLAR program translation.

When talking about context implication and the estimation of $P(d \rightarrow q)$, we have to be aware that many different interpretations of the implication probability are possible (Crestani and Lalmas, 2001). We apply two interpretations which are based on the view of the content of a context as a *conjunction* or *disjunction* of propositions (Rölleke, 2008). In the former case, all propositions in the query must also be contained in the document, whereas in the latter case, it is sufficient if only one proposition is contained in both the query and the document. In the probabilistic case, every common proposition would raise the implication probability. We base the following discussion on terms, but our arguments can be applied to other propositions (categorisations, attributes) as well.

6.2.4.1 Conjunctive Interpretation

The conjunctive interpretation of contexts regards a context as a conjunction of its propositions. For example, the context d1 and also the query q1 above would be interpreted as `ir` $\wedge$ `db`. A document

```
d2[ ir  !db ]
```

would be interpreted as `ir` $\wedge\neg$ `db`.

What does $d \rightarrow q$ mean then? In the non-probabilistic case, it means that the Boolean expression of q is contained in d's expression so that from d being true it follows that q is true as well. In other words: each proposition in q is contained in d.

In the probabilistic case, $P(d \rightarrow q)$ is the probability that this holds. Following Wong and Yao (1995), indexing estimates the degree to which a document is relevant to a concept (term), and this is denoted $P(d \rightarrow t)$. On the other hand, $P(t \rightarrow q)$ estimates to which degree a concept is relevant to the query. In our framework, we then set $P(d \rightarrow t) = P(\texttt{term(}t\texttt{,}d\texttt{)})$ and $P(t \rightarrow q) = P(\texttt{term(}t\texttt{,}q\texttt{)})$. If we assume each proposition in a context as being independent, the implication probability can be computed (for terms) as

$$\begin{aligned} P(d \rightarrow q) &= \prod_{t \in q} P(d \rightarrow t) \cdot P(t \rightarrow q) \\ &= \prod_{t \in q} P(\texttt{term(}t\texttt{,}d\texttt{)}) \cdot P(\texttt{term(}t\texttt{,}q\texttt{)}) \end{aligned}$$

To express this in FVPD, we model `implies` as a conjunction based on the propositions found in the query. For the above example, we would get

```
1 implies(D,q1) :- term(ir,q1) & term(ir,D) &
2                  term(db,q1) & term(db,D)
```

For d1, this is computed as

$$\overbrace{0.6}^{\texttt{term(ir,q1)}} \cdot \overbrace{0.6}^{\texttt{term(ir,d1)}} \cdot \overbrace{0.9}^{\texttt{term(db,q1)}} \cdot \overbrace{0.8}^{\texttt{term(db,d1)}} = 0.2592$$

One of the concepts discussed in Section 5.3.2.2 are predicate spaces, which reflect the importance of a predicate (a term, an attribute or a category). They can be based, e.g., on the inverse document frequency of a term, an attribute or a category. Many IR systems create a ranking incorporating the inverse document frequency (*idf*) and the within term frequency (*tf*). If we assume $P(\texttt{term(}t\texttt{,}d\texttt{)})$ to be based on the frequency of a term t in a document d, and if we further model the term space with the relation `termspace` so that the weight of `termspace(`t`)` is using the inverse document frequency of t in the collection, we can extend `implies` so that it creates a $tf \times idf$-based ranking. For our example, we get

```
1 implies(D,q1) :- term(ir,q1) & term(ir,D) & termspace(ir) &
2                  term(db,q1) & term(db,D) & termspace(db)
```

Let us suppose $P(\texttt{termspace(ir)}) = 0.5$ and $P(\texttt{termspace(db)}) = 0.2$. The evaluation of the rule for d1 yields:

$$\overbrace{0.6}^{\texttt{term(ir,q1)}} \cdot \overbrace{0.6}^{\texttt{term(ir,d1)}} \cdot \overbrace{0.5}^{\texttt{termspace(ir)}} \cdot \overbrace{0.9}^{\texttt{term(db,q1)}} \cdot \overbrace{0.8}^{\texttt{term(db,d1)}} \cdot \overbrace{0.2}^{\texttt{termspace(db)}} = 0.02592$$

If knowledge augmentation is requested ("//D->q"), we replace `term` with `term_k` for documents in a rule for `implies_k`:

```
1 implies_k(D,q1) :- term(ir,q1) & term_k(ir,D) & termspace(ir) &
2                    term(db,q1) & term_k(db,D) & termspace(db)
```

With knowledge augmentation enabled, we ask if the augmented context of document d implies the query. The probability for this is higher because new terms might be introduced to d as a result of the augmentation.

The conjunctive view ensures that only documents are returned which are indexed with *all* query term or propositions. Documents which do not know about all query propositions are not retrieved. This is similar to Boolean queries with an AND conjunction. Only few documents are retrieved, usually with low probabilities.

As already mentioned, the conjunctive interpretation ensures that only documents are returned which contain all query terms. While for some applications this might be desirable, it might be too strict for others, even when relaxed by knowledge augmentation. Furthermore, the conjunctive interpretation calculates very low probabilities, even for documents which might be highly relevant. Additionally, from a strict theoretic point of view, the rules presented above are problematic, because we would expect that lower weights in the query context lead to higher implication probabilities. For example, a document containing the term 'ir' but not 'db' would also imply q1 with a certain probability, since the probability that q1 contains 'ir' is only 0.6, which leads to a 0.4 probability that 'ir' is *unknown* in q1. In a strict sense, the lower the probability that 'ir' is *true* in q1, the higher the probability that a document containing 'db' but not 'ir' implies q1. To solve this in the rules above, we re-interpret the term weights in the query context as the degree of importance of query terms, expressed as $P(t \rightarrow q)$. Another disadvantage of the conjunctive view comes from a computational point of view. Since conjunctions in FVPD are usually implemented as Join operations, the processing of `implies` is computationally expensive. Furthermore, `implies` must be newly created for every query q by analysing q's propositions and building an according subgoal list for `implies`.

6.2.4.2 Disjunctive Interpretation

The disjunctive interpretation of contexts sees the content of a context as a disjunction of its propositions. For instance, the context d1 and the query q1 above would be interpreted as `ir` $\vee$ `db`. The document d2 would be interpreted as `ir` $\vee\neg$ `db`.

When interpreting a context's content in a disjunctive way, $d \rightarrow q$ means that at least one proposition of q must be contained in d so that from d being true it follows that q is true as well. The rule

```
1 implies(D,Q) :- term(T,D) & term(T,Q)
```

reflects this. When evaluated, `T` is substituted with every term. As soon as there is one term contained in both `q1` and a document, the implication becomes true for the corresponding document. The probability that this happens is computed by means of the inclusion-exclusion formula (see Def. 24 on page 133). For our example document d1, $P($`implies(d1,q1)`$)$ is

$$\overbrace{0.6}^{\texttt{term(ir,q1)}} \cdot \overbrace{0.6}^{\texttt{term(ir,d1)}} + \overbrace{0.9}^{\texttt{term(db,q1)}} \cdot \overbrace{0.8}^{\texttt{term(db,d1)}} - \overbrace{0.6}^{\texttt{term(ir,q1)}} \cdot \overbrace{0.6}^{\texttt{term(ir,d1)}} \cdot \overbrace{0.9}^{\texttt{term(db,q1)}} \cdot \overbrace{0.8}^{\texttt{term(db,d1)}} = 0.8208$$

Again we can extend `implies` so that it creates a $tf \times idf$-based ranking. This is done by adding a `termspace` subgoal:

```
1 | implies(D,Q) :- term(T,D) & term(T,Q) & termspace(T)
```

The calculation of the implication probability `implies(d1,q1)` for our example document then yields:

$$\overbrace{0.6}^{\texttt{term(ir,q1)}} \cdot \overbrace{0.6}^{\texttt{term(ir,d1)}} \cdot \overbrace{0.5}^{\texttt{termspace(ir)}} + \overbrace{0.9}^{\texttt{term(db,q1)}} \cdot \overbrace{0.8}^{\texttt{term(db,d1)}} \cdot \overbrace{0.2}^{\texttt{termspace(db)}} -$$
$$\overbrace{0.6}^{\texttt{term(ir,q1)}} \cdot \overbrace{0.6}^{\texttt{term(ir,d1)}} \cdot \overbrace{0.5}^{\texttt{termspace(ir)}} \cdot \overbrace{0.9}^{\texttt{term(db,q1)}} \cdot \overbrace{0.8}^{\texttt{term(db,d1)}} \cdot \overbrace{0.2}^{\texttt{termspace(db)}} = 0.29808$$

To support other propositions like categorisations and attributes, we extend `implies` accordingly to

```
1 | implies(D,Q) :- term(T,D) & term(T,Q) & termspace(T)
2 | implies(D,Q) :- instance_of(O,C,D) & instance_of(O,C,Q) &
3 |                 categoryspace(C)
4 | implies(D,Q) :- attribute(A,O,C,D) & attribute(A,O,C,Q) &
5 |                 attributespace(A)
```

The relation `categoryspace` (`attributespace`) describes the importance of a certain category (attribute), which can be determined by the inverse document frequency of the category or attribute, respectively. If knowledge augmentation should be applied, `implies` becomes

```
1 | implies_k(D,Q) :- term_k(T,D) & term_k(T,Q) & termspace(T)
2 | implies_k(D,Q) :- instance_of_k(O,C,D) & instance_of_k(O,C,Q) &
3 |                   categoryspace(C)
4 | implies_k(D,Q) :- attribute_k(A,O,C,D) & attribute_k(A,O,C,Q) &
5 |                   attributespace(A)
```

From a processing point of view, the disjunctive view has some advantages over the conjunctive interpretation. Above rules are generic, they do not need any knowledge about the query to be created, so we do not have to rebuild them for a new query. Furthermore, the number of Join operations is limited. `implies`, as we implemented it here, makes the interpretation of $P(d \rightarrow q)$ a symmetric measure. $P(d \rightarrow q)$ and $P(q \rightarrow d)$ both yield the same result. An alternative would be to consider the implementation of the vector space model, as it is reported in (Rölleke, 1998, section 6.3), which would calculate different values for $P(d \rightarrow q)$ and $P(q \rightarrow d)$. However, although we are able to express this interpretation in FVPD, the evaluation of the resulting rules is computationally complex; we therefore do not further discuss this option here, but refer to (Rölleke, 1998, section 6.3) for details.

Because `implies` estimates the probability that a document implies a query, and we do not ask for the probability that it does not, we can assume a closed world for the `implies` relation.

6.2.4.3 Translation Rules for Implications

We have to extend the translation of subgoals in Section 6.2.2.1 to cover implications as well:

$$\begin{aligned} \textit{trans}(\text{subgoal,c}) &:= \texttt{implies(c1,c2)} \\ &\quad \text{if subgoal} = \texttt{c1->c2} \\ &:= \texttt{implies\_k(c1,c2)} \\ &\quad \text{if subgoal} = \texttt{//c1->c2} \end{aligned}$$

Note that the translation of the implication might also involve the creation of suitable `implies` or `implies_k` rules in case of a conjunctive interpretation.

In our above considerations we used term, classification (categorisation) and attribute predicates. These predicates reflect the importance of the given term, category or attribute. As POLAR allows us to specify them directly, we choose the following translation:

$$\begin{aligned} \textit{trans}(\text{predicate}) &:= \text{weight } \textit{trans}(\text{term-predicate}) \\ &\quad \text{if predicate} = \text{weight term-predicate} \\ &:= \text{weight } \textit{trans}(\text{class-predicate}) \\ &\quad \text{if predicate} = \text{weight class-predicate} \\ &:= \text{weight } \textit{trans}(\text{attr-predicate}) \\ &\quad \text{if predicate} = \text{weight attr-predicate} \\ \textit{trans}(\text{term-predicate}) &:= \texttt{termspace(t)} \\ &\quad \text{if term-predicate} = {}^{\circ}\texttt{t} \\ \textit{trans}(\text{class-predicate}) &:= \texttt{categoryspace(c)} \\ &\quad \text{if term-predicate} = {}^{\circ\circ}\texttt{c} \\ \textit{trans}(\text{attr-predicate}) &:= \texttt{attributespace(a)} \\ &\quad \text{if term-predicate} = {}^{\circ\circ\circ}\texttt{a} \end{aligned}$$

We assume a closed world for predicates, so we have to add:

```
1 _CWA(termspace)
2 _CWA(categoryspace)
3 _CWA(attributespace)
```

6.2.5 Relevance Augmentation

One of POLAR's main concepts is knowledge augmentation. With knowledge augmentation, we are able to take into account the annotation and structural context of a document for retrieval by propagating propositions and their probabilities from subcontexts to their corresponding supercontexts. But there may be cases where we cannot perform knowledge augmentation. Consider, for instance, the scenario where an independent annotation service is connected to several digital library management systems (DLMS) which handle the main documents, as it is illustrated in Agosti et al. (2006) for the DiLAS service. Such an annotation service might have access to annotations, (annotation) fragments and annotation types (plus their polarity), but not to the full texts of the documents managed by the connected digital library management services. But at least for annotation-based document search it would be crucial to fetch the document-only retrieval status value in order to bias it later with the values coming from annotations. So we need a result combination, similar to the one reported in Agosti and Ferro (2005), where retrieval services provided by the single DLMSs return retrieval weights for its managed documents, which are then to be merged with the retrieval status values of the annotations (which are calculated within the annotation service). One possible solution for POLAR

to handle such cases is *relevance augmentation*. The main idea of relevance augmentation is to draw conclusions about the relevance of a supercontext from the relevance of its subcontexts by propagating retrieval status values. This idea is inspired by the spreading activation approach as described e.g. in Frei and Stieger (1994). An early version of relevance augmentation, which is formulated in FVPD, too, and also takes into account the effect of negative annotations, is introduced in Frommholz et al. (2004b). This approach is based on annotation types explicitly given as discourse structure relations as they are used in the COLLATE annotation model. Positive annotations are used to raise the probability that the annotated object is relevant, while negative annotations decrease this probability. We follow this philosophy in our approach as well.

Support Rules for Relevance Augmentation Similar to knowledge augmentation, relevance augmentation is implemented by providing a set of support rules which define the intensional relation `relevant_k`. The evaluation of `relevant_k` post-processes the retrieval weights of annotations, determined by the corresponding `implies` rules, with the retrieval status values of documents, externally delivered by the DLMSs and stored in the `relevant_external` relation. This makes relevance augmentation a two-phase process: first, the content-based retrieval weight is calculated, then this weight is used to generate the final, context-based result. The FVPD support rules are as follows:

```
1  # Phase 1: Initial content-based weight
2  relevant(D,Q) :- relevant_external(D,Q).
3  relevant(A,Q) :- implies(A,Q).

5  # Phase 2: Context-based re-weighting
6  relevant_k(D,Q) :- relevant(D,Q).
7  relevant_k(D,Q) :- relevant_k_logical(D,Q).
8  relevant_k(D,Q) :- relevant_k_posanno(D,Q).
9  relevant_k(D,Q) :- relevant_k_reference(D,Q).
10 !relevant_k(D,Q) :- relevant_k_neganno(D,Q).

12 relevant_k_logical(D,Q) :- acc_subpart(D,S) & relevant_k2(S,Q).
13 relevant_k_posanno(D,Q) :- acc_mtarget(D,S) & relevant(S,Q).
14 relevant_k_posanno(D,Q) :- acc_fragment(D,S) & relevant(S,Q).
15 relevant_k_posanno(D,Q) :- acc_canno(D,S)  & relevant_k2(S,Q).
16 relevant_k_neganno(D,Q) :- acc_negcanno(D,S) & relevant_k2(S,Q).
17 relevant_k_reference(D,Q) :- acc_reference(D,S) & relevant_k2(S,Q).



20 relevant_k2(D,Q) :- relevant(D,Q).
21 relevant_k2(D,Q) :- relevant_k2_logical(D,Q).
22 relevant_k2(D,Q) :- relevant_k2_posanno(D,Q).
23 relevant_k2(D,Q) :- relevant_k2_reference(D,Q).
24 !relevant_k2(D,Q) :- relevant_k2_neganno(D,Q).

26 relevant_k2_logical(D,Q) :- acc_subpart(D,S) & relevant_k2(S,Q).
27 relevant_k_anno(D,Q) :- acc_fragment(D,S) & relevant(S,Q).
28 relevant_k2_anno(D,Q) :- acc_canno(D,S)  & relevant_k2(S,Q).
```

```
29 relevant_k2_neganno(D,Q) :- acc_negcanno(D,S) & relevant_k2(S,Q).
30 relevant_k2_reference(D,Q) :- acc_reference(D,S) & relevant_k2(S,Q).
```

Lines 2 and 3 implement the initial content-based phase. Line 2 uses the (possibly normalised) retrieval status value delivered by external DLMSs for documents – this RSV is the weight of a corresponding `relevant_external` fact. In line 3, we use `implies`, which calculates an RSV for annotations based on the implication probability determined by the propositions in the query and the annotation (as discussed in Section 6.2.4). Note that we do not apply any knowledge augmentation, since we are going to propagate relevance values later; using knowledge augmentation as well would mean to consider subcontexts twice. The rule in line 6 says that the content-based relevance weight is considered. Similar to knowledge augmentation, we extend the context with subcontexts coming from the logical document structure (if known), annotations and referenced objects. This is reflected in rules 7 to 10. Note that the rule in line 10 deals with negative evidence coming from annotations. The rules in lines 6 – 10 therefore collect positive and negative evidence about the relevance of a document. Line 12 realises the document structure-based augmentation through the subpart relation. Lines 13 to 16 implement annotation-based relevance augmentation in a similar manner as for knowledge augmentation. In these lines, we collect both positive (l. 13 – 15) and negative evidence (l. 16) about relevance. In our approach, we have negative evidence for relevance if the document accesses a negative annotation which is deemed relevant. The rule in line 17 propagates the relevance weight of referenced objects.

As discussed for knowledge augmentation, we also do not consider meta annotations for relevance augmentation. Furthermore, merged targets are only considered when they belong to the context to augment, and subcontexts of fragments are not considered. Therefore we need a second rule set, like we needed it for knowledge augmentation, which can be seen in lines 20 – 30 and which works analogously to the first rule set.

Example 17 (Relevance Augmentation): To illustrate the effect of relevance augmentation, consider the example in Section 4.2.6.3 and especially Figure 4.3 on page 70. With relevance augmentation, the POLAR query "`?- //D->q1`" is translated to "`?- relevant_k(D,q1)`" in FVPD. For the query q1 and a document d1, an external DLMS returns $P($`relevant_external(d1,q1)`$) = 0.3$. The POLAR instance calculates $P($`implies(a1,q1)`$) = 0.5$ and $P($`implies(a2,q1)`$) = 0.2$. The access probabilities from d1 to a1 and a2, respectively, are $P($`acc_canno(d1,a1)`$) = P($`acc_negcanno(d1,a2)`$) = 0.5$. The relevance augmentation support rules combine these three retrieval weights and generate a new context-based one for d1. Lines 6 to 9 collect the positive evidence coming from d1, so we infer with a probability of $0.3 + 0.5 \cdot 0.5 - 0.3 \cdot 0.5 \cdot 0.5 = 0.475$ that d1 is relevant ($P($`relevant_k(d1,q1)`$) = 0.475$). Considering the negative evidence from the annotation a2, we infer that d1 is not relevant with $0.5 \cdot 0.2 = 0.1$ probability ($P($`!relevant_k(d1,q1)`$) = 0.1$). The evaluation of "`?- relevant_k(d1,q1)`" would combine positive and negative evidence and calculate the probability that we have positive evidence and no negative evidence from the context[9], that is $0.475 \cdot (1 - 0.1) = 0.4275$, which is the final context-based retrieval status value of d1. □

The considerations make clear that we should assume an open world for `relevant_k`, since we collect evidence for relevance and non-relevance in parallel here. But in order to

[9] Recall from Section 6.1.2 that `relevant_k(D,Q)` is translated into the PD expression `pos_relevant_k(D,Q) & !neg_relevant_k(D,Q)`.

combine positive and negative evidence from annotations correctly, the four truth values for events in `relevant_k` must not be disjoint since otherwise, only positive evidence would be considered[10].

When performing relevance augmentation, we again need to take measures in case there are cycles. To solve this, serialisation could be applied just as we did it for knowledge augmentation. Another option would be to perform radius-1 relevance augmentation. Since relevance augmentation can only be performed during query time, radius-1 relevance augmentation, where we only consider direct subcontexts, is probably the preferred choice, as it is more efficient.

The design of the relevance augmentation rules allows for focusing on selected subcontext types, exactly as for knowledge augmentation. So we could easily perform relevance augmentation with or without the annotation context but only the structural one, for instance. An interesting variant of relevance augmentation is including data from relevance feedback – if a user judged documents or annotations relevant w.r.t. a query, than this knowledge can be applied together with relevance augmentation to modify the initial ranking incorporating the feedback data.

6.3 System Architecture and Java Implementation

6.3.1 POLAR Translation and Execution

Figure 6.1 shows the UML class diagram of the main classes in POLAR. POLAR programs are executed with the `HyPOLAREngine`. This engine offers a method `eval` which takes the actual POLAR program as parameter and returns a ranked list (not further specified here). During processing, a POLAR program is first translated into FVPD and then evaluated (executed) with an FVPD engine.

6.3.1.1 Translation

The `POLARTranslator` class is responsible for the translation of POLAR to FVPD. Its `translate` method takes a POLAR program (an instance of `HyPOLARProgram`) as parameter and returns the FVPD translation (an instance of `HyFVPDProgram`) according to the given FVPD dialect. The POLAR program is parsed with a lexer and parser (`POLARLexer` and `POLARParser`). The parsing process transforms POLAR programs into an intermediate syntax tree using (not further specified) classes for basic POLAR objects, which are propositions, components (subparts), annorefs, fragments, merged targets, references, predicates, rules and queries. The intermediate format is then translated into FVPD using an implementation of the `FVPDDialect` interface. We specify `FVPDDialect` as an interface to allow for the easy integration of new translation rules, which we call *dialects* here. For example, two different implementations of `FVPDDialect` might offer different interpretations of the `implies` rules, so different FVPD dialects would return different translations of POLAR programs into FVPD. Other FVPD dialects might lay the focus on certain optimisations or provide alternative rules for knowledge or relevance augmentation (and additionally provide means to fetch RSVs from external DLMSs in order to set up the `relevant_external` relation). Each implementation of `FVPDDialect` must offer methods to translate the basic POLAR objects in the syntax tree to FVPD. The `trans` methods either take a basic POLAR object and its context, a rule,

[10] If the tuples in `relevant_k` were disjoint, then $P($`relevant_k(d,q)`$)$ in FVPD would be $P($`pos_relevant_k(d,q) & !neg_relevant_k(d,q)`$) = P($`pos_relevant_k(d,q)`$)$ on the PD level, so no negative evidence would be considered.

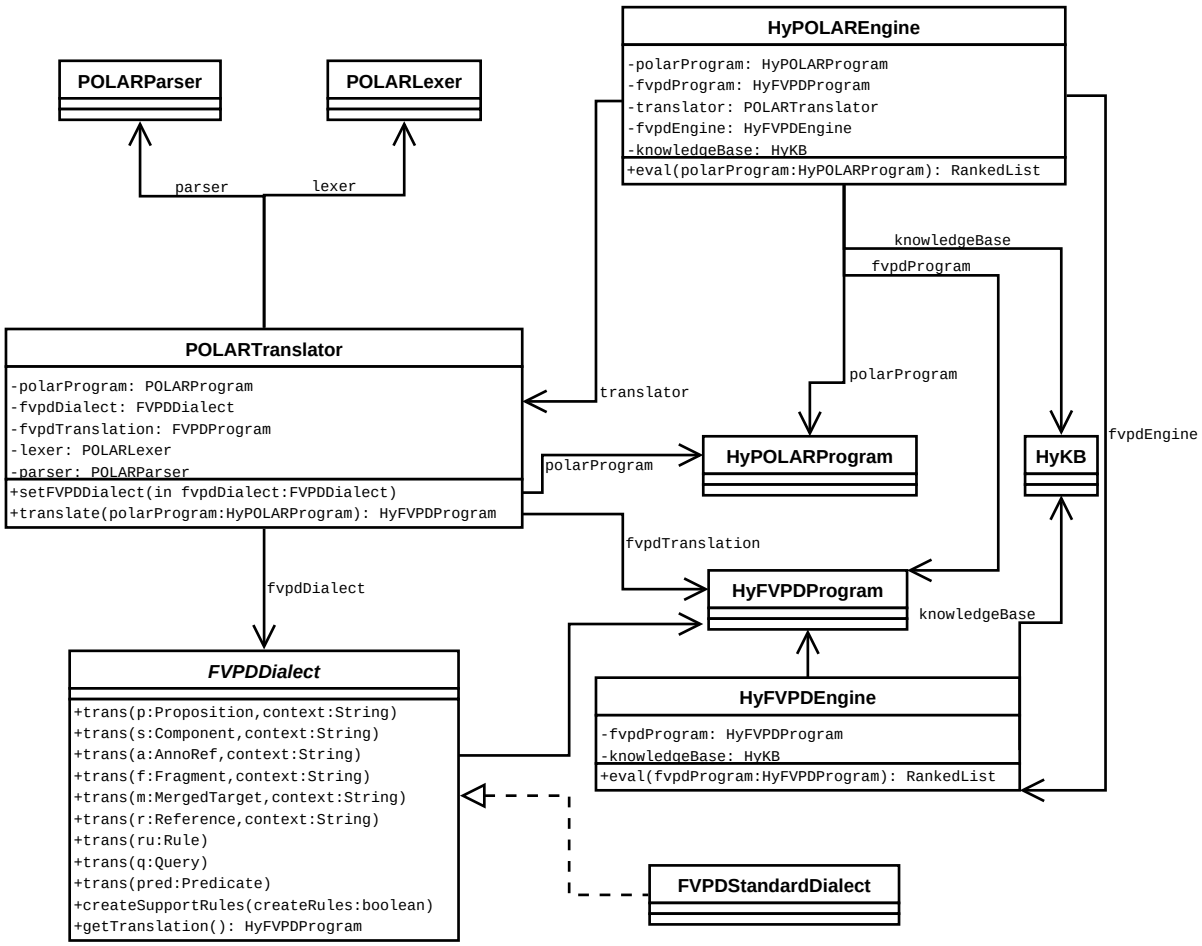


Figure 6.1: POLAR translation and execution classes as UML class diagram

a query or a predicate. With the `createSupportRules` method it can be specified if the support rules for knowledge augmentation should be created or not. `getTranslation` returns a `HyFVPDProgram` object representing the FVPD translation, together with the required support rules, if specified.

6.3.1.2 Evaluation

Once the POLAR program is translated, it is evaluated with `HyFVPDEngine`. This engine takes the FVPD program together with an instance of `HyKB` as parameters. Any instance of `HyKB` holds information about the underlying knowledge base, which contains the extensional relations consisting of indexed propositions (terms, classifications and attributes) within documents and annotations, the access relations and their type, as well as information about the term, attribute and categorisation spaces. It might further contain indexed augmented contexts in case knowledge augmentation was applied during indexing time. To evaluate the FVPD program based on the current knowledge base, the `HyFVPDEngine` instance invokes the FVPD engine provided by HySpirit. It parses the output of the FVPD engine and returns a ranked result list to the invoking `HyPOLAREngine`, which in turn propagates the ranked list to its invoking application.

6.3.2 POLAR Indexing

Any instance of `HyPOLAREngine` executes POLAR programs which contain, besides rules and queries, contexts (documents, annotations, fragments, merged targets), their content (propositions) and their relations to other objects in the structured annotation hypertext. Expressing the knowledge base as a POLAR program is feasible if it contains only a few objects, like all the POLAR examples provided in the previous chapters. However, if we talk about thousands or even millions of objects, such a knowledge base would be by far too large to represent them as a POLAR program. We also need sophisticated indexing structures to provide fast access for retrieval. It is therefore crucial to store the FVPD relations for basic POLAR knowledge modelling (as described in Section 6.2.1) as well as rules and even the new relations created by knowledge augmentation (like `term_k`), which can be produced offline, in an internal indexing structure holding the whole static knowledge base. The `HyKB` object contains all information about the knowledge base; this knowledge base is linked to the POLAR program executed by `HyPOLAREngine`. This way, a POLAR program might consist of few query contexts, some further rules and POLAR queries. The information about documents, annotations, fragments, merged targets, components, their content and their interrelations, is stored in the index represented by the corresponding `HyKB` object. During execution of the POLAR program, the `HyKB` object is linked to the POLAR program and used by the `HyFVPDEngine` object which is responsible for the actual execution of the generated FVPD code.

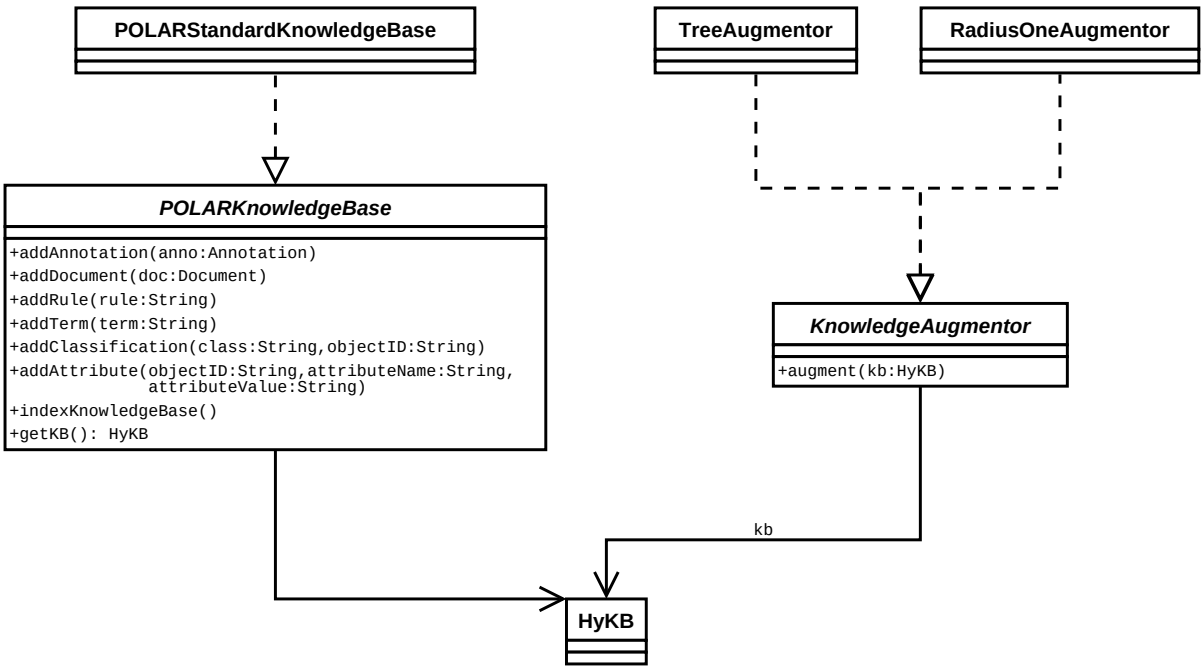


Figure 6.2: POLAR indexing classes as UML class diagram

POLAR provides classes for indexing which return the required `HyKB` object. Figure 6.2 shows the main classes used for the creation of a POLAR index and knowledge base. Most important here are two interfaces, `POLARKnowledgeBase` and `KnowledgeAugmentor`. The first one is used to build the POLAR index, while the latter one is responsible for offline knowledge augmentation and the creation of a persistent `term_k` relation.

POLARKnowledgeBase is designed as an interface because it should support representations which need to be consistent to the applied FVPDDialect implementation. This means that the FVPD program produced by FVPDDialect must be able to use the internal structure and relations created by the corresponding POLARKnowledgeBase object. Furthermore, different implementations of the interface might store the knowledge base in a database or in the file system. Annotations and documents (and their content and relations to other objects) are added to the knowledge base with the addAnnotation and addDocument methods. For convenience, addAnnotation and addDocument take Annotation and Document objects, respectively, as parameter. These objects also embed fragments and merged targets. Further methods of POLARKnowledgeBase manage rules and predicates (term, attribute and predicate spaces). Once all objects are added, the indexKnowledgeBase methods can be invoked, which indexes (or updates, in case incremental indexing is supported by an implementing class) the objects and creates the relations and structure contained in the knowledge base. This also involves applying stemming and stop word elimination as well as calculating (possibly four-valued) proposition weights. getKB returns the HyKB object representing the knowledge base.

An implementation of KnowledgeAugmentor takes a HyKB object as parameter and performs offline knowledge augmentation on all contexts contained in the knowledge base. Offline knowledge augmentation creates a static version of the term_k (attribute_k, instance_of_k, respectively) relation which is then stored in the knowledge base. If this persistent version of term_k is used, no support rules for augmentation should be generated by the FVPDDialect object.

6.3.3 POLAR Prototype

The current POLAR prototype[11] is implemented in Java using a special Java API for HySpirit called JaySpirit[12]. POLARLexer and POLARParser are based on the ANTLR parser generator[13]. They extend ANTLR's Lexer and Parser object, respectively. The prototype further provides the classes POLARTranslator, HyPOLAREngine and HyPOLARProgram. HyFVPDProgram, HyFVPDEngine and HyKB are part of the JaySpirit package. HyFVPDEngine invokes the HySpirit engine when evaluating a HyFVPDProgram object, using the knowledge base specified in the HyKB object.

The POLAR prototype contains an implementation of the FVPDDialect interface called FVPDStandardDialect (see Fig. 6.1). This class implements the translation of POLAR to FVPD for basic knowledge modelling, queries and rules, as reported in sections 6.2.1 and 6.2.2. If requested, non-serialised support rules for knowledge augmentation as shown in Section 6.2.3.1 are created. These rules propagate both probabilities for *true* and *false* for terms, attributes and classifications. The class translates the expression "D->q" to the $tf \times idf$-based disjunctive retrieval function implies_k in Section 6.2.4.2.

The class POLARStandardKnowledgeBase implements the POLARKnowledgeBase interface and creates the index and knowledge base compatible to FVPDStandardDialect. POLARStandardKnowledgeBase stores the knowledge base in the file system and uses a probabilistic relational index (Rölleke, 2008), which is provided by the HySpirit distribution, for fast access.

Another part of the POLAR prototype are classes that realise offline knowledge augmentation with certain restrictions. The TreeAugmentor class implements the KnowledgeAugmentor

[11] http://www.is.inf.uni-due.de/projects/polar/index.html.en
[12] http://www.is.inf.uni-due.de/projects/jayspirit/index.html.en
[13] http://www.antlr.org/

interface. It uses the serialised knowledge augmentation rules presented on page 162, but assumes the underlying access structure to be a tree. Another implementation of `KnowledgeAugmentor` is `RadiusOneAugmentor`. The class realises Radius-1 augmentation as discussed on p. 165. Both `TreeAugmentor` and `RadiusOneAugmentor` propagate probabilities of a term to be *true*, but do not handle *false* as well as categorisations and attributes yet. The resulting `term_k` relation is written into a knowledge base which is compatible to `POLARStandardKnowledgeBase`.

6.4 Summary and Discussion

In this chapter we presented an implementation of the POLAR framework based on four-valued probabilistic Datalog. We chose FVPD since one of its distinctive features is that it can handle POLAR's four truth values and deal with an open world assumption (but a closed world can be declared for certain relations). FVPD programs are translated into probabilistic Datalog ones, so in essence, POLAR can utilise existing PD implementations like HySpirit. To implement POLAR with FVPD, a translation function is defined. Access relations and their type (the special propositions in Section 5.2 denoting if a relation represents a subpart, an annotation or a reference) are mapped onto appropriate two-valued predicates in FVPD. Any other POLAR proposition and its weight list is mapped onto four-valued predicates for terms, classifications and attributes, which also contain the context the proposition belongs to as parameter.

POLAR queries and rules are translated into corresponding FVPD queries and rules by giving a translation for each head and subgoal. The translation of subgoals can be controlled in case knowledge augmentation is requested.

For knowledge augmentation, a set of rules can be defined which propagate probabilities that a proposition is *true* or *false* from subcontexts to its supercontext in order to introduce new propositions in the supercontext or modify the weight of existing ones. The FVPD rules which control knowledge augmentation need to be attached to any FVPD translation of POLAR programs. When performing knowledge augmentation, we have to handle cycles which may occur in the access structure. This can be done by applying a serialisation approach (which needs to analyse the access structure) or a restriction to radius-1 augmentation. The knowledge augmentation approach can easily be adapted to include or exclude certain access types; for example, only the logical document structure (subparts) or the annotation context (merged targets, fragments and content annotations) may be subject to augmentation. Another advantage of the knowledge augmentation approach is that it can be performed during indexing.

Possible POLAR retrieval functions estimate the probability $P(d \rightarrow q)$ that a document d implies a query q. Many such estimations are conceivable, which might be based on a conjunctive or disjunctive interpretation of documents and annotations. Our example retrieval functions produce a $tf \times idf$-based ranking. Besides the propositions in the given context, these functions also consider the term, category and attribute space, respectively. Retrieval functions are implemented in POLAR by linking a set of helper rules which describe the intensional `implies` relation.

Once a retrieval status value is calculated as an estimation of $P(d \rightarrow q)$, we can apply another augmentation strategy called relevance augmentation. In contrast to knowledge augmentation, we calculate a content-only retrieval weight for each context first and then augment each context by propagating the retrieval weights from subcontexts to supercontexts. The newly computed context-based retrieval status value combines positive and negative evidence found in the accessed subcontexts. The big advantage of relevance augmentation is that it integrates the

results from external retrieval services, which is important if an annotation service employing POLAR does not have access to the full texts of external sources in order to index them, but can call the sources's search routines through an API. One of the disadvantages of relevance augmentation is that it can only be applied during query time, so radius-1 augmentation might be preferred. Similar to knowledge augmentation, relevance augmentation is implemented by providing FVPD support rules.

The POLAR system is designed to flexibly integrate and apply different translation strategies which must be implemented in accordance to the `FVPDDialect` interface. Each FVPD dialect can implement different augmentation strategies or retrieval functions. This way we preserve the flexibility we gain by applying the FVPD framework in the current POLAR implementation – new augmentation and retrieval strategies can easily be plugged into the current implementation by just providing according FVPD dialects. For indexing and offline augmentation, the interfaces `POLARKnowledgeBase` and `KnowledgeAugmentor` are defined. The current POLAR prototype is written in Java and provides an implementation of the discussed classes and interfaces.

How does POLAR's implementation compare to the POOL one? The POLAR translation into FVPD is naturally similar to the POOL implementation reported in Rölleke (1998). However, some differences exist and some new concept were introduced in POLAR. The main difference between POLAR and POOL lies in the implementation of the augmentation rules. POOL's knowledge augmentation rules had to be refined in order to deal with the peculiarities of the different access types. In contrast to POOL, the access structure in POLAR can contain cycles, so we introduced a serialisation strategy to handle them. We also discussed possibilities for radius-1 augmentation. As in POOL, retrieval functions in POLAR are based on probabilistic inference. This makes POOL's retrieval functions potential candidates for POLAR. Additionally to POOL's retrieval functions implementing the vector space model, we presented $tf \times idf$-like retrieval functions based on an explicit conjunctive and disjunctive view on documents and queries. POLAR's concept of relevance augmentation is so far unknown to POOL.

Translating POLAR programs into FVPD is one possibility to implement POLAR according to the semantics presented in the last chapter. The advantage of the above translation rules is that they provide an intuitive mapping of POLAR expressions to FVPD ones. However, there might be equivalent translations which are less intuitive but more efficient. For example, we could split the `term` relation into several ones reflecting the different context types – a relation `fragment_term(t1,f1)` might say that term t1 is contained in a fragment f1. This makes queries for fragment terms more efficient, since it would not be required to search the whole `term` relation. A knowledge augmentation rule like

```
term_k(T,D) :- acc_fragment(D,S) & term(T,S).
```

would turn into

```
term_k(T,D) :- acc_fragment(D,S) & fragment_term(T,S)
```

whose evaluation would be more efficient. Similar relations can be defined for other subcontext types (e.g., `term_canno` for terms in content annotations).

Another option addressing efficiency is translating POLAR programs directly into PD or PRA programs. The current translation pipe is POLAR → FVPD → PD → PRA, which restricts our options to create efficient PRA programs (which are the ones executed in the end). Our system

architecture might be extended to allow for the direct translation of POLAR programs into PD or PRA by introducing PD or PRA dialects and defining and implementing a corresponding `PDDialect` or `PRADialect` interface. An example where the evaluation of POLAR programs can benefit from the direct translation into PD is the propagation of inconsistent knowledge, as we have seen in Section 6.2.3.1. Another example is *relaxed knowledge augmentation* (Rölleke, 2008). Let us consider the FVPD knowledge augmentation rule

```
term_k_anno(T,D) :- acc_fragment(D,S) & term(T,S).
```

for propagating terms in fragments to the corresponding supercontext. According to Section 6.1.2, this is translated to the PD rule

```
pos_term_k_anno(T,D) :-
        acc_fragment(D,S) & pos_term(T,S) & !neg_term(T,S).
```

The evaluation of the negation "`!neg_term(T,S)`" is computationally expensive, so we may decide for a relaxed strategy which does not contain the negation, but uses the PD rule

```
pos_term_k_anno(T,D) :- acc_fragment(D,S) & pos_term(T,S).
```

instead. This variant would also propagate inconsistent knowledge and does not work exactly as we described knowledge augmentation in the last chapter, but it is more efficient. Unfortunately, this PD rule cannot be expressed in FVPD and would thus benefit from a direct translation of POLAR into PD and corresponding execution with a PD engine.

We introduced retrieval functions based on the notion of probabilistic inference, which compute the probability $P(d \rightarrow q)$ that a document implies a query (and the other way round). We indicated above that we are not restricted to this particular retrieval method, but can use the expressive power of FVPD, PD or even PRA to introduce new retrieval approaches to annotation-based IR within POLAR. For example, Rölleke et al. (2007) shows possible alternative retrieval functions expressed in PD and based on well-known concepts like, e.g., language models. These retrieval approaches might be integrated into POLAR as distinctive FVPD dialects or even on the PD or PRA level.

In the last chapters we discussed the POLAR framework, the concepts and data structures behind it, and its syntax, semantics and implementation. The next part of this thesis deals with the question whether certain annotation-based POLAR concepts lead to more effective retrieval results than established retrieval approaches.

Part III

Evaluation

7

Example Applications and Test Collections

Or to some coffee-house I stray,
For news, the manna of the day,
And from the hipp'd discourses gather
That politics go by the weather.

(Matthew Green)

In the last part the POLAR framework was proposed; its core retrieval functionality is based on augmentation and probabilistic inference. The goal of this part is to answer the question whether or not annotations improve retrieval effectiveness. To do so, we first need a suitable collection of documents and annotations, and second a set of example topics/queries with corresponding relevance judgements which state whether a document or annotation is relevant to the given query or not. While we learn from the discussion in Chapter 2 that the annotation universe comprises a broad range of possible applications, there is the problem of finding such a suitable test collection for the evaluation of annotation-based IR. This fact recently led to the proposal of the automatic creation of such a collection from existing ones (Agosti et al., 2007b).

In order to evaluate the annotation-based retrieval methods in POLAR, two collections were chosen: an email discussion list from the World Wide Web Consortium (W3C) and a snapshot from ZDNet News. While the first collection, which is suitable for discussion search, was created collaboratively within an evaluation initiative, we took a seven month snapshot of ZDNet News for the latter one and created the required topics and relevance judgements on our own in order to set up a collection for annotation-based document search. In this chapter, the collections are presented, we describe how testbeds were created and how the collection can be represented in POLAR. In the next chapter, experiments and their results are described.

7.1 Emails as Annotations: The W3C Discussion Lists

7.1.1 Collection

The W3C lists are part of a crawl of the public W3C sites (`*.w3.org`) in June 2004. This crawl was the official collection of the Enterprise Track at the Text Retrieval Evaluation Conference (TREC) in 2005 and 2006. The Enterprise Track was divided in several subtasks, among which discussion search was one (Craswell et al., 2005). The collection contains the W3C email lists

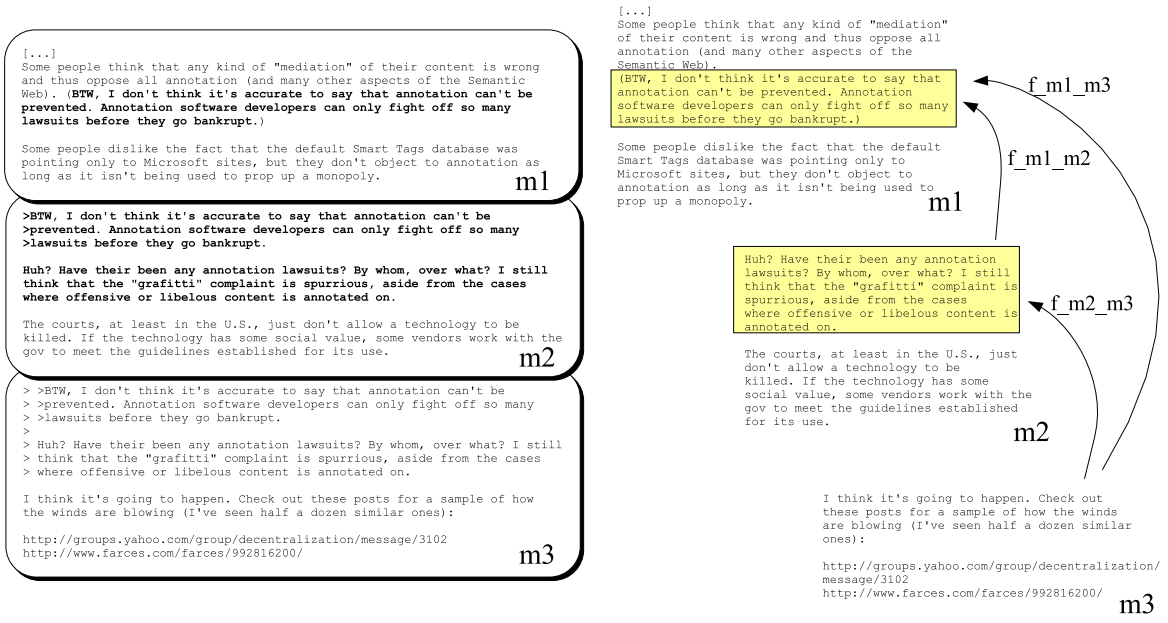


Figure 7.1: Example for the annotation view on email discussions

where topics related to the World Wide Web Consortium are discussed. From the provided HTML representation of the email lists, 174,307 emails were extracted. In the TREC 2005 runs, which are the relevant ones here, the participants submitted 59 topics which were considered for the final set. For the assessments, each topic was assigned to two of the participating groups – first to the group who suggested the topic, second to another group. For document judgements, there were three types of answers: not relevant to the query, topically relevant, relevant with a pro/con statement. The last answer was used for experiments where systems should not only find out if a message was topically relevant, but also if it contained a pro or contra statement. Subject to judgement was the new part of an email; the quotations or replies were ignored. An email is thus relevant only if its new part is relevant. For our experiments, we only used the judgements whether a message was topically relevant or not; we did not explicitly address the problem of finding pro/con statements.

7.1.2 The Annotation View on Email Messages

The left hand side of Figure 7.1 shows some example email messages m1, m2 and m3, which were taken from a W3C discussion list. m2 replies to m1, and m3 is a reply to m2. Replies, such as m2 and m3, usually contain two different parts in their body:

- the *quotations*, which are passages of the original text. Quotations are identified by quotation characters which prefix each line of the quotation[1]. Such a quotation character can be '>' or ':' or probably other characters or regular expressions; combinations of them usually define the *quotation depth*. In m3, quotations belonging to m1 have the depth 2 and are identified by two quotation characters ('> >'), whereas quotations belonging to m2 are identified by the single quotation character '>';

[1]Unfortunately, there are exceptions to this rule.

- the *unquoted* or *new* part containing new content by the author of the message.

In replies, the unquoted (or new) part contains annotations (here: textual, shared comments) of passages of previous messages. These passages are determined by the corresponding quotations. Quotations are thus the annotation targets. This *annotation view on email messages* (Frommholz, 2005a) is illustrated on the right hand side of Fig. 7.1. The quotation in m2 defines the fragment f_m1_m2 the new part in m2 belongs to. Due to different quotation depths we can divide the quoted part in m3 – one fragment (f_m2_m3) actually belongs to m2, but the part with the higher quotation depth, f_m1_m3, even belongs to m1. So the new part in m3 relates to two parts in two different previous postings, as the arrows in Fig. 7.1 indicate. The transformation of emails into a POLAR representation thus includes parsing quotations and assigning the fragments to their original objects (f_m1_m2 and f_m1_m3 to m1, f_m2_m3 to m2), determined by the quotation depth. The merged annotation target of m2, q_m2, is made of f_m1_m2, while the merged annotation target q_m3 contains f_m1_m3 and f_m2_m3, the whole quotation of m3. This example also shows that annotations can belong to more than one object.

Emails usually consist of several quotations and new parts following each other, for example:

```
>> line1
> line2
line3

>> line4
> line5
line6
```

Each new part of the email is seen as an annotation of the corresponding quotation (`line3` as an annotation of `line1` and `line2`, `line6` as an annotation of `line4` and `line5`). In the experiments for the TREC Enterprise track (presented in the next chapter), the task was to find emails whose new part is relevant to the query, but in the representation above, we are dealing with parts of emails – when applying discussion search, we would get a retrieval status value for each `line3` and `line6`, but we are interested in one retrieval status value for the whole email, comprising all new parts. We therefore combine all parts of the email by merging all quotations and all new parts, so as a result we gain one big quotation and one big new part for each email. In our example, the outcome of the restructuring is:

```
>> line1
>> line4
> line2
> line5
line3
line6
```

The corresponding fragment of the predecessor's predecessor would consist of `line1` and `line4`, while the fragment belonging to the direct predecessor would be `line2` and `line5`. `line3` and `line6` establish the new part of the example email. In this view, the annotation consists of `line3` and `line6`, annotating `line1`, `line4`, `line2` and `line5`.

7.1.3 Collection Statistics

After setting up the collection as discussed above, the index contained 174,307 emails. 79,853 of them have a merged annotation target[2]. 35,963 have at least one fragment, and 49,611 a direct content annotation. As shown in Fig. 7.2, most emails with fragments (25,302) have 1 fragment; 7,186 have 2 fragments and 2,181 emails 3 fragments. 25 emails have 10 or more fragments, 1 email has the highest number of fragments (19). On average, each email with fragments has 1.46 fragments. There are 33,300 threads, and 38,392 emails are thread leaves (i.e., they do not have any reply). In Fig. 7.3 we can see that most of the 49,611 emails with

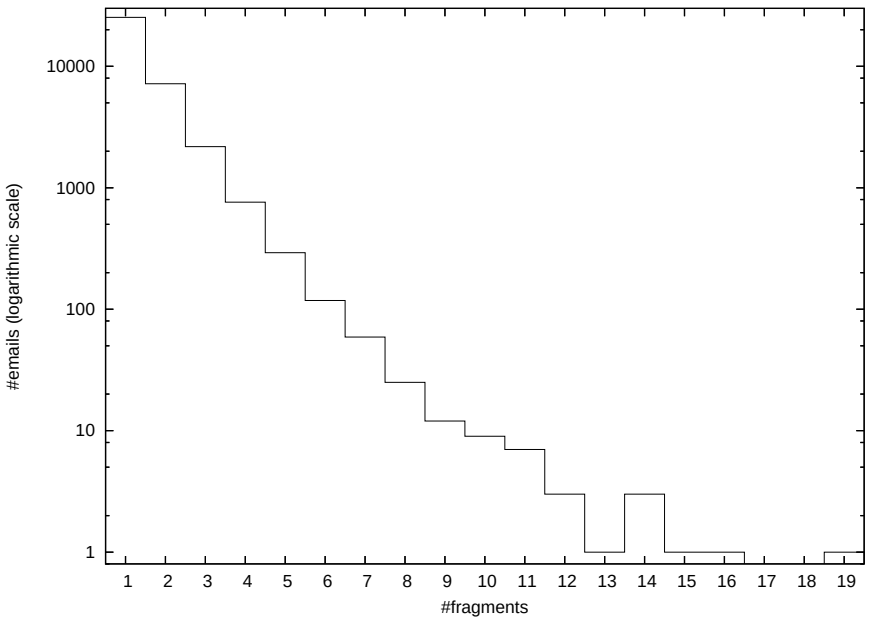


Figure 7.2: W3C Lists: emails and fragments

direct comments, 39,948, have 1 annotation. 7,948 emails have 2 comments (or replies), 1,335 have 3 comments. Only 2 emails have more than 10 comments. Emails with replies have 1.24 comments on average.

7.1.4 Representation in POLAR

We take the discussion thread in Fig. 7.1 as an example. The root of an email discussion thread does not reply to any other email (in theory) and can therefore not be regarded as an annotation in our model. To reflect this, we introduce new classes $\mathsf{Message} \sqsubseteq \mathsf{AnnotatableObject}$ and $\mathsf{Reply} \sqsubseteq \mathsf{Annotation} \sqcap \mathsf{Message}$ and categorise our messages accordingly:

```
annotation(A) :- reply(A)
message(M) :- reply(M)
annotatableObject(E) :- message(E)
message(m1)   reply(m2)   reply(m3)
```

Message m1 can be represented in POLAR as follows (we only consider a part of the term set of messages in the example; all probabilities are fictitious):

[2]Also full quotes, i.e. emails which quote the whole previous message, are counted here.

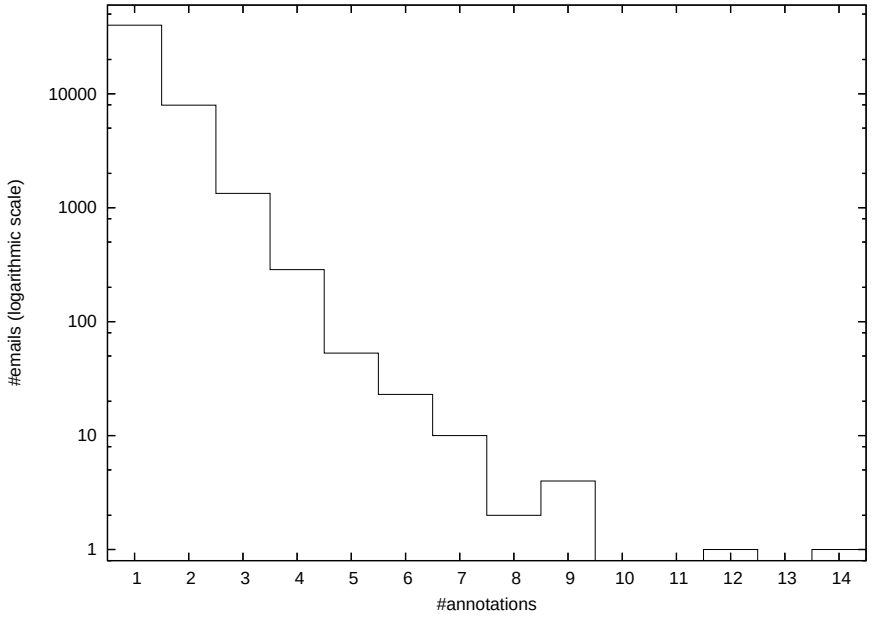


Figure 7.3: W3C Lists: emails and annotations

```
m1[ 0.5 mediation   0.6 annotation   0.5 lawsuit
    0.6 people   0.4 database
    0.3 f_m1_m2|| 0.7 annotation   0.6 lawsuit   0.2 *m2 ||
    0.3 f_m1_m3|| 0.7 annotation   0.6 lawsuit   0.2 *m3 ||
  ]
```

This says that the body of m1 consists of terms like 'mediation', 'annotation', 'lawsuit', 'people' and 'databases'[3]. A fragment of m1 containing the terms 'annotation' and 'lawsuit' is quoted by two other emails, m2 and m3 (and is therefore their annotation target). The probability that we access the context f_m1_m2 or f_m1_m3 to determine the augmented context m1(f_m1_m2) or m1(f_m1_m3), respectively, is 0.3. m2 and m3 can be represented as

```
m2[ 0.9 q_m2< 0.75 annotation 0.7 lawsuit >
    0.55 annotation 0.5 lawsuit 0.3 graffiti
    0.4 courts 0.6 technology
    0.3 f_m2_m3|| 0.8 annotation   0.75 lawsuit
                  0.6 graffiti   0.2 *m3 ||
  ]

m3 [ 0.9 q_m3< 0.85 annotation   0.8 lawsuit
               0.5 graffiti >
     0.4 happen   0.6 posts   0.3 sample   0.55 winds
   ]
```

The quotation of m2, q_m2, contains the terms 'annotation' and 'lawsuits', whereas the quotation of m3, q_m3, additionally contains the term 'graffiti'. q_m3 is a merged annotation target from f_m1_m3 and f_m2_m3.

[3]We attached the title of the message to its body. An alternative would be to regard the title as a subpart of a message, for instance "`m1[title1[...]]`"

7.2 ZDNet News

ZDNet News[4] is a web site that delivers news and developments in information technology and business. After registering, users can write comments on ZDNet articles, which again can be commented. This way, discussion threads are created which are attached to its root article. The HTMLish nature of the comments also allows for referencing external documents via annotations.

7.2.1 Collection Statistics

We downloaded a snapshot of ZDNet News and set up a testbed for our experiments based on this collection. Figure 7.2.1 shows an example thread of comments belonging to an article. The ZDNet snapshot was harvested from December 2004 to July 2005. It consists of 4,704

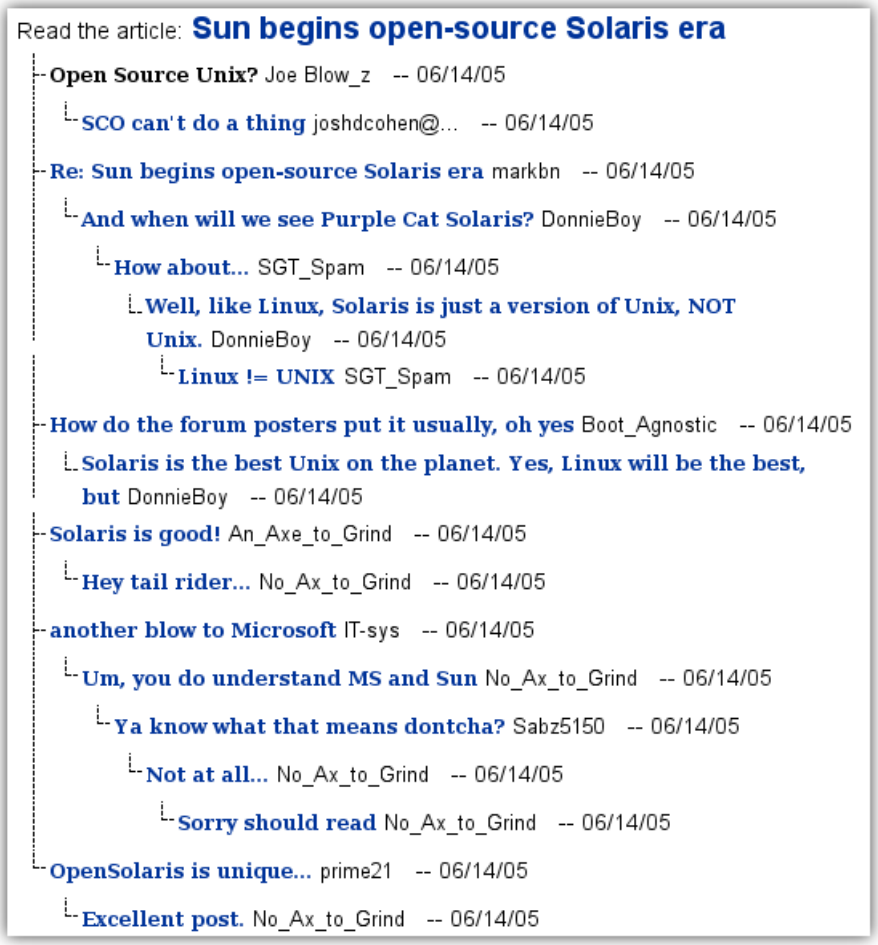


Figure 7.4: ZDNet article and discussion thread

articles and 91,617 annotations, from which 26,107 are direct comments on the articles. 3,486

[4] http://news.zdnet.com/

articles have at least one comment, and articles with comments have 7.4891 comments on average. As we can see from the histogram in Figure 7.5, most documents only had a few direct comments. 761 articles have 1 direct comment, 534 articles 2, and 2,778 articles have less than 10 comments. 52 articles have more than 50 direct comments; among them 17 have more than 100 direct comments and 3 have even more than 200. One article has the highest number of comments, 262.

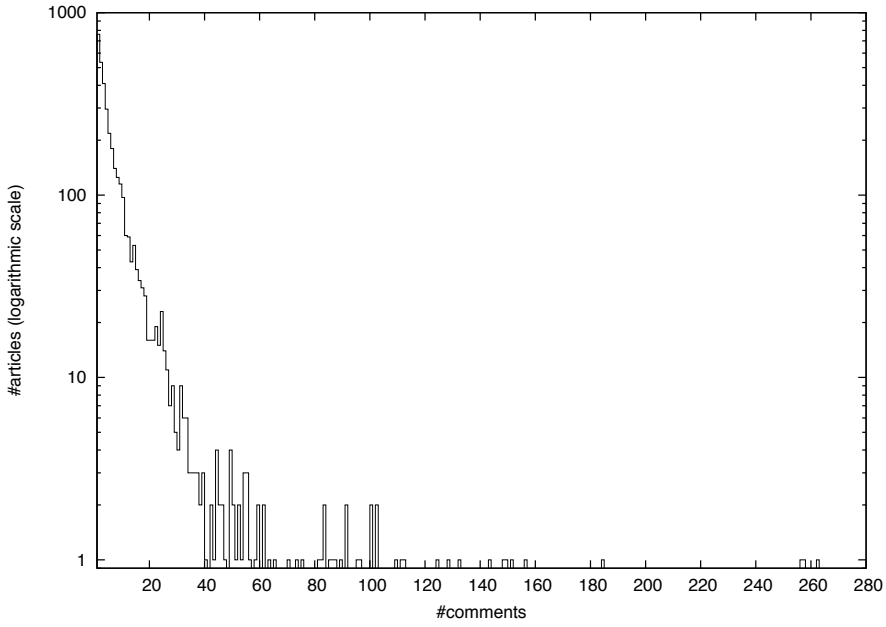


Figure 7.5: ZDNet News snapshot: articles and direct annotations

7.2.2 Testbed Creation

To create our testbed, 20 topics and queries were defined. Since relevance assessments could not be achieved within an evaluation initiative like INEX or TREC, we only had limited resources for the assessments – we asked colleagues, students and IT experts outside our institute to assess the topics. As our assessors volunteered for working on the topics in their spare time, we asked them to assess 150 documents (articles) per topic. To create an initial ranking, a simple approach for annotation-based document search was applied: articles and their direct annotations were merged and regarded as one atomic document[5]. The resulting ranking was used for the relevance judgements.

A 3-tier ranking system was defined. The assessment procedure was as follows: after reading an article, the assessor looks if it is *relevant* w.r.t. the given topic. If so, it is judged like that. If not, the assessor looks at the direct annotations to see if there are relevant comments. If the assessor finds any relevant comment, the article is judged as being *not relevant but having relevant annotations*. If there are also no relevant comments, the article is judged as *not relevant*. Non-relevant articles might nevertheless be interesting to users when they are indirectly relevant, i.e. there are annotations which contain the information the users seek (and the system is able to point them to these). A similar relevance criterion is reported for

[5]Since the POLAR prototype was not available at the date the assessments started, no pooling procedure was performed. The initial ranking was produced directly with HySpirit.

Web documents in Hawking et al. (1999). The scenario supports the ranking enriched with annotation showcase in Section 4.3.2 on page 74. From all documents judged, the assessors classified 679 articles as being relevant and 113 as not relevant but having relevant annotations.

The testbed was also used as a subcollection for the INEX Heterogeneous Track (Frommholz and Larson, 2007).

7.2.3 Representation in POLAR

The ZDNet commenting system does not provide formal means to relate a comment to certain fragments of a message, nor does it allow for annotating more than one target. This means that no fragments could be extracted. Furthermore, as we set up the collection for annotation-based document search, it would not make sense to create merged annotation targets (which would consist of content of the whole annotated object). All comments are categorised as being content level annotations to the object they refer to, although some of them might be better regarded as meta level annotations. Furthermore, the titles of articles and comments were merged with their respective body[6]. Additional information (like classifications/tagging of articles) was not used. This makes the representation of ZDNet articles and comments in POLAR quite straightforward, as the following simple example (with fictitious term weights and access probabilities, extracted from the example in Fig. 7.2.1) shows:

```
document(zdnet_art1)
annotation(zdnet_com1)
annotation(zdnet_com2)
annotation(zdnet_com1_1)
annotation(zdnet_com2_1)
zdnet_art1[ 0.7 sun   0.6 open-source   0.8 solaris   0.4 era ...
            0.4 *zdnet_com1   0.4 *zdnet_com2 ...
          ]
zdnet_com1[ 0.3 open   0.5 source   0.6 unix ...
            0.4 *zdnet_com1_1 ...
          ]
zdnet_com2[ 0.6 sun   0.4 open-source   0.75 solaris   0.3 era ...
            0.4 *zdnet_com2_1
          ]
zdnet_com1_1[ 0.8 sco   0.6 thing ...
            ]
zdnet_com2_1[ 0.8 purple   0.8 cat   0.85 solaris ...
             0.4 *zdnet_com2_1_1
            ]
...
```

7.2.4 Polarity of Comments

One of the peculiarities of the ZDNet News snapshot is that we find many controversial discussions there. Unfortunately, ZDNet does not provide any means to mark a comment as being positive or negative to the object it replies to, so one of the experiments reported in the next

[6] Again, titles could be regarded as part of the logical document structure and be a subpart of the article or comment.

chapter dealt with the question whether it is possible to automatically detect the polarity of a comment accurately. To perform such an experiment, another testbed based on the ZDNet News collection was created. Assessors classified comments into six sentiment categories: positive, negative and neutral, all three on the content and meta level:

Content:Positive Positive sentiment on the content level, usually expressed with phrases like "I agree". 140 comments were judged so.

Content:Negative Negative sentiment on the content level, expressed with phrases like "I disagree". 327 comments were judged as being negative on the content level.

Content:Neutral Neutral sentiment on the content level, the author does not explicitly express any sentiment. 502 comments were categorised into this class.

Meta:Positive Positive sentiment on the meta level, for example by stating how useful a a comment was ("this helped me a lot"). 33 comments were judged so.

Meta:Negative Negative sentiment on the meta level; a classical example in discussion forums are so-called "trolls", which are persons whose comments usually do not convey any useful information. The usual reaction to such comments is a phrase like "don't feed the trolls!". 104 comments were regarded as being negative on the meta level.

Meta:Neutral No sentiment expressed on the meta level. 135 comments were classified as being neutral on the meta level.

Multiple classification was possible, e.g. if a comment was half positive and half negative, both a positive and negative category could be chosen. If we do not distinguish between meta and content level, 173 comments were judged as being positive, 637 neutral and 431 negative. This results to 1,241 judgements given in total by 10 assessors (colleagues and students).

A procedure which automatically classifies comments into the categories positive and negative might for instance detect that zdnet_com1_1 (the comment with the title "SCO can't do a thing" in Fig. 7.2.1) is negative; the above POLAR representation could then be modified to

```
...
zdnet_com1[ 0.3 open   0.5 source   0.6 unix ...
            0.4 -⋆zdnet_com1_1 ...
          ]
...
zdnet_com1_1[ 0.8 sco   0.6 thing ... ]
...
```

so that zdnet_com1_1 is a negative content annotation now. From a retrieval point of view, the information about the sentiment of a comment could be exploited to determine the trustworthiness of an annotation, as it is outlined in the showcase in Section 4.3.8 on page 79. Furthermore, the fact that zdnet_com1_1 influences the propagation of its terms in the knowledge augmentation process; the terms 'sco' and 'thing', for instance, would contribute to the probability that they are negative in the augmented context zdnet_com1(zdnet_com1_1).

7.3 Summary and Discussion

In this chapter, two collections and their modelling in POLAR were presented. These are the W3C mailing lists, which were part of the collection used for the TREC Enterprise Track, and ZDNet News consisting of IT-related articles and discussion threads. The former collection can be used for a discussion search task, while with the latter one, experiments on annotation-based document search can be performed. Since a collection containing documents and annotations was not available yet, example queries and relevance judgements had to be collected first. Based on the ZDNet snapshot, another testbed was created to perform experiments which aim at the classification of comments into positive and negative ones, which is a prerequisite for example to determine the trustworthiness of annotations and influences the knowledge augmentation process. Again, the required judgements needed to be collected.

Are the W3C lists a good test collection for annotation-based discussion search? In Bottoni et al. (2003), newsgroup postings and email messages are not regarded as annotations. They see quotations as part of the reply rather than of the message they belong to, and consequently argue that a reply does not relate to a portion of the previous message. This is the main difference to the annotation view on emails above, where we take a different viewpoint and see quotations as a location attribute. One might further argue against the annotation view on email messages that there is no real "main document" involved which is annotated, as is the case with paper documents and also forums which usually refer to a main document whose content is discussed. We might on the other hand regard the root of a discussion thread as the document which sets the topic for the further discussion and is thus the "main document". This does not contradict to the definition and usage of annotations we have seen so far – replying to a previous message usually means making critical remarks or explanatory notes, and it can be regarded as a case of active reading. Another argument for the annotation view on email messages is that annotation-based discussion can indeed be realised applying a similar technical solution ("email-like" textual annotations with quotations and unquoted parts) as used for newsgroup or mailing list postings. Furthermore, the task of discussion search focuses on annotations rather than on annotated documents, so the lack of a "real" root document does not play a role here.

In contrast to the W3C lists, the snapshot of ZDNet News contains a "real" root document, namely the ZDNet article an annotation thread belongs to. This makes the ZDNet snapshot a potential candidate for experiments on annotation-based document search. However, due to limited resources it was only possible to create and judge 20 topics. According to a recent study (Sanderson and Zobel, 2005), this spoils the quality of the test collection, but should nevertheless be sufficient to provide at least preliminary results.

We have seen in Chapter 2 that there is a broad range of annotations, systems, and annotation tasks. Our collections above cover only a few of them, namely public discussions about certain topics, which follow no certain purpose, and the detection of the polarity of annotations. However, annotation systems and the act of annotation might have a specific goal and specific users in mind. In the COLLATE system, for example, a closed group of scholars used annotations to interpret the material at hand. The fact that experts are writing the annotations naturally raises their quality compared to open systems like ZDNet News. On the other hand, collections like ZDNet News are a suitable playground for evaluating methods which determine the trustworthiness of public annotations. Another example are private annotations, as they are offered by systems like DAFFODIL, or reviews and ratings. It is clear that results of experiments with the collections above cannot necessarily be transferred to other annotation tasks,

but they can nevertheless gain valuable insights for the tasks and situations they cover. It is obvious that future work should consider the creation of new annotation-based test collections. The experiments discussed in the next chapter can only be the beginning.

8

Experiments

A statistician can have his head in the oven and his feet in ice, and on average he feels fine.

(Anonymous)

In the last chapter, two test collections and their POLAR representation were introduced: W3C email discussions and ZDNet News. The first one was used for discussion search experiments, while with the latter one, annotation-based document search as well as a machine learning approach to determine the polarity of annotations were evaluated. This chapter presents and discusses the experiments and their results. We begin with an introduction of the applied methodology and evaluation measures. Subsequently, term weighting and retrieval functions are presented. Afterwards, results of discussion and document search experiments are shown and discussed. The chapter ends with a general reflection of the results.

8.1 Methodology and Presentation

8.1.1 Evaluation Measures

In the experiments, typical measures known from information retrieval are applied, which are recall and precision. Both values are used to measure the *effectiveness* of a retrieval system, which addresses the quality of the produced ranking. In contrast to the effectiveness, the *efficiency* measures how many system resources (CPU time, memory) are used for retrieval. Since our implementation depends on the HySpirit system and its efficiency, we do not evaluate this, but only the effectiveness of the described annotation-based retrieval approaches. For evaluation, the `trec_eval` program written by Chris Buckley[1] was used. We briefly summarise the main measures used in the experiments.

Definition 26 (Evaluation measures):
Recall denotes the ratio of relevant documents which are found by the system to the whole number of relevant documents in the repository. Let RET be the set of retrieved

[1] `http://trec.nist.gov/trec_eval/`; see the README file in the package for further explanations on the evaluation measures. We used version 7.3 for the evaluation.

documents and REL be the set of relevant documents in the system. The recall r is then defined as

$$r := \frac{|RET \cap REL|}{|REL|}$$

All relevant documents are found by a system if $r = 1$.

Precision is the ratio of relevant documents found to the number of retrieved documents. Reusing the denotations above, the precision p is

$$p := \frac{|RET \cap REL|}{|RET|}$$

All retrieved documents are relevant if $p = 1$.

From the basic measures recall and precision, new measures can be derived. For a query, it is common to measure the precision at 11 recall levels $r = 0.0, 0.1, 0.2, \ldots, 1.0$. For example, the recall level 0.1 is the point where 10% of the relevant documents were found; here, p can be interpolated as the maximum of precision at all recall points $>= 0.1$. These values can be displayed in *recall-precision graphs*, where the recall points are the x-axis and the corresponding precision the y-axis; precision is thus plotted as a function of recall. Some interpolation strategy might be applied to connect the precision values between the recall points. From the precision values at certain recall points, `trec_eval` calculates an *average precision* as the arithmetic mean of the precision obtained after each relevant document was found. All values so far are related to one single topic. A retrieval run usually consists of several defined topics, so we are interested in measures reflecting all topics. For recall-precision curves, a macro-averaging strategy is applied which calculates the precision average over all topics at each recall point. For N topics, we get

$$p_r^{avg} := \frac{1}{N} \sum_{i=1}^{N} \frac{|RET_{i,r} \cap REL|}{|RET_{i,r}|}$$

at each of the recall points $r = 0.0, 0.1, 0.2, \ldots, 1.0$; $RET_{i,r}$ is the set of documents retrieved at recall point r for topic i. Furthermore, the *mean average precision* (MAP) is the mean of the average precision values of all topics.

Measuring the precision at pre-defined recall levels is a bit artificial, although a good system-oriented measure. If we assume users only examine the first K ranked documents, we are also interested in the precision after K documents are retrieved. We denote this precision as $P@K$; for example, P@10 is the precision after 10 documents are retrieved. Again, theses values are averaged when considering all queries.

8.1.2 Significance Tests

To compare a ranking produced by a system w.r.t. a given baseline system and evaluation measure, paired t-tests with confidence $p < 0.05$ and $p < 0.01$ (Sanderson and Zobel, 2005) were performed. The following null hypothesis was proclaimed:

$$H_0 : \quad \mu_1 - \mu_2 = 0$$

with μ_1 as the mean of the baseline sample and μ_2 as the mean of the sample from the system to compare to the baseline. A sample consists of values for single topics. For instance, to test the significance of the results for two systems w.r.t. MAP, two samples for each systems are examined. These samples consist of MAP values for each topic. So the null hypothesis to evaluate is that both the baseline system and the system to evaluate are equally effective with respect to the given measure, or, in other words, that the sample values for both systems come from the same population (Clauß and Ebner, 1972). A statistical significant difference ($p < 0.05$ at least) leads us to reject H_0. The alternative hypothesis is as follows:

$$H_1: \quad \mu_1 - \mu_2 \neq 0$$

which means that the differences to the baseline are significant.

8.1.3 Presentation

In the tables below, we mark statistical significant and non-significant results as follows:

Significance	Example
Not significant	0.25
Significant ($p < 0.05$)	*0.25*
Very significant ($p < 0.01$)	*0.25*

Furthermore, the best results achieved w.r.t. to a certain measure are printed in bold.

8.2 Term Weighting and Retrieval Functions

In the experiments, the following weighting function for term weights within contexts (documents and annotations) was used:

Definition 27 (Within-context term weights):
Consider a term t appearing in a context d. In POLAR, t is assigned a probability $p_{t,d}$ depending on the term frequency of t in d (so we get "`d[`$p_{t,d}$ `t]`" as the POLAR representation). We calculate $p_{t,d}$ as

$$p_{t,d} = \frac{tf(t,d)}{avgtf(d) + tf(t,d)}$$

with $tf(t,d)$ as the frequency of term t in document or annotation d and $avgtf(d)$ as the average term frequency of d, calculated as

$$avgtf(d) = \sum_{t \in d^T} tf(t,d)/|d^T|$$

and d^T being the set of terms occurring in document d.

We also use global weights for terms as follows:

Definition 28 (Global term weights):
We assign a weight p_t to each term t, denoted in POLAR as "p_t °t". p is computed based on the inverse document frequency as

$$p_t = idf(t)/maxidf$$

with

$$idf(t) = -\log\left(df(t)/numdoc\right),$$

$df(t)$ as the number of objects (documents and annotations) in which t appears, $numdoc$ as the number of objects in the collection and $maxidf$ is the maximum inverse document frequency.

In all experiments, the $tf \times idf$-based interpretation of `D->q` discussed in Section 6.2.4.2 was used.

8.3 Discussion Search

In this section we present some experimental results for discussion search. First results for discussion search were achieved while participating in the TREC 2005 Enterprise Track[2] (Frommholz, 2005a) and performing subsequent experiments (Frommholz and Fuhr, 2006b,a). The collection used are the W3C email lists introduced in Section 7.1, consisting of 174,307 emails and 59 test queries. Note that the initial discussion search runs published in (Frommholz, 2005a) were repeated, which produced different results.

As discussed, emails provide us with a rich structure from which we can extract fragments and merged annotation targets. Therefore the main target of discussion search evaluation was to examine the role of fragments and merged annotation targets when exploited for knowledge augmentation[3].

8.3.1 Description of Runs

For the description of the runs, we recall the example from Section 7.1.4; the message m2 contains all elements we want to evaluate:

```
m2[ 0.9 q_m2< 0.75 annotation 0.7 lawsuit >
    0.55 annotation 0.5 lawsuit 0.3 graffiti
    0.4 courts 0.6 technology
    0.3 f_m2_m3|| 0.8 annotation   0.75 lawsuit
                  0.6 graffiti     0.2 *m3 ||
  ]
```

[2] http://www.ins.cwi.nl/projects/trec-ent/

[3] Note that in Frommholz and Fuhr (2006a), also experiments for relevance augmentation are reported. These generated basically worse results than the knowledge augmentation runs. In the hindsight, the relevance augmentation experiments turned out to be not expressive. The different results between knowledge and relevance augmentation were mainly due to an inappropriate extensional evaluation of the relevance augmentation rules (with intensional evaluation, they would have produced the same result as knowledge augmentation). We therefore do not discuss the relevance augmentation results any further.

We will use different POLAR representations of m2 for illustration. The following runs with specific parameter settings were performed:

Baseline In the baseline run, we only looked at the new parts of emails. All quotations were removed, no knowledge augmentation performed. In the annotation view, this is equivalent to considering only the annotation content for retrieval of relevant annotations. The example message m2 and the query would be modelled in POLAR as

```
m2[ 0.55 annotation 0.5 lawsuit 0.3 graffiti
    0.4 courts 0.6 technology                ]
?- D->q
```

Whole Email The whole email is regarded as a document. We do not distinguish between quoted and new parts here. In the annotation view, this would be the same as merging the content of annotation targets with the annotation's content. With this strategy, m2 and the query would be expressed in POLAR as

```
m2[ 0.65 annotation 0.6 lawsuit 0.3 graffiti
    0.4 courts 0.6 technology                ]
?- D->q
```

(note the slightly higher probabilities for 'annotation' and 'lawsuit' due to their appearance in the quoted part).

knowlaug-fragment-<acc_prob> In these runs, knowledge augmentation is performed which only considers the extracted fragments (see Section 7.1.4). <acc_prob> denotes the corresponding access probability; for instance, the run "knowlaug-fragment-0.1" means knowledge augmentation with fragments and access probability 0.1. In POLAR, m2 and the query would then be

```
m2[ 0.55 annotation 0.5 lawsuit 0.3 graffiti
    0.4 courts 0.6 technology
    0.1 f_m2_m3|| 0.8 annotation   0.75 lawsuit
                  0.6 graffiti ||               ]
?- //D->q
```

knowlaug-mtarget-<acc_prob> Knowledge augmentation with merged targets. Again, <acc_prob> denotes the corresponding access probability; for instance, the run "knowlaug-mtarget-0.7" means knowledge augmentation with merged targets and access probability 0.7. A POLAR example is

```
m2[ 0.7 q_m2< 0.75 annotation 0.7 lawsuit >
    0.55 annotation 0.5 lawsuit 0.3 graffiti
    0.4 courts 0.6 technology                ]
?- //D->q
```

knowlaug-canno-<acc_prob> Knowledge augmentation with content annotations. <acc_prob> is the probability that we access content annotations. In POLAR, m2, m3 and the query are (with an access probability of 0.4):

```
m2[ 0.55 annotation 0.5 lawsuit 0.3 graffiti
    0.4 courts 0.6 technology   0.4 *m3      ]
m3[ 0.4 happen 0.6 posts 0.3 sample 0.55 winds]
?- //D->q
```

Note that in these runs, only radius-1 augmentation (see page 165) was performed.

knowlaug-all-<c_acc_prob>-<f_acc_prob>-<m_access_prob> In these runs, knowledge augmentation is performed with all evidence (coming from merged targets, fragments and content annotations). <c_acc_prob> is the content annotation access probability, <f_acc_prob> is the fragment access probability, while <m_access_prob> denotes the merged target access probability. If, for instance, we access content annotations with a probability of 0.4, fragments with 0.1 probability and merged targets also with a probability of 0.1, m2, m3 and the query would be represented in POLAR as:

```
m2[ 0.1 q_m2< 0.75 annotation 0.7 lawsuit >
    0.55 annotation 0.5 lawsuit 0.3 graffiti
    0.4 courts 0.6 technology
    0.1 f_m2_m3|| 0.8 annotation   0.75 lawsuit
                  0.6 graffiti
                  0.4 *m3 ||                    ]
m3[ 0.4 happen 0.6 posts 0.3 sample 0.55 winds]
?- //D->q
```

(recall fragment permeability – this is equivalent to "`m2[... 0.4 *m3]`"). For content annotations, radius-1 augmentation was performed again.

8.3.2 Baseline and Whole Email Results

Table 8.1 shows the results of the Baseline and Whole Email runs. Both strategies could easily be applied outside POLAR. The Whole Email run used the information in the quotations (the merged targets in our model). It performed better than the baseline, although a statistically significant improvement could be gained only w.r.t. the mean average precision. The results

Run	MAP	P@5	P@10	P@15	P@20	P@30
Baseline	0.2834	0.4915	0.4475	0.4102	0.3822	0.3333
Whole Email	*0.3125*	0.5017	0.4746	0.4192	0.3941	0.3492

Table 8.1: Mean average precision (MAP) and precision at K documents retrieved (P@K) of Baseline and Whole Email runs

give a hint that the context of an annotation, given by the content of the objects it annotates, is an important further evidence for its relevance.

8.3.3 Results for Knowledge Augmentation

8.3.3.1 Merged Annotation Targets

In a first series of experiments, the effect of merged targets to retrieval effectiveness is evaluated. In the annotation view on emails, merged targets actually are the quoted part of the email.

The question is: does retrieval effectiveness benefit from performing knowledge augmentation with merged targets/quotations?

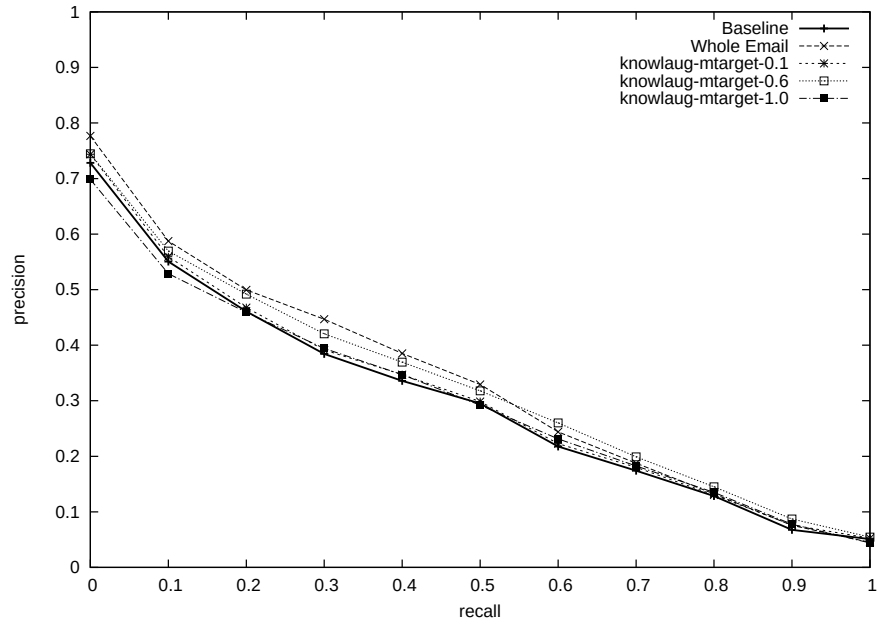


Figure 8.1: Recall-precision graph of selected knowledge augmentation merged target runs

Figure 8.1 shows the recall-precision-graph of selected merged target runs, together with the baseline and the Whole Email runs. The recall-precision curves of further experiments can be seen in Appendix C.2. Knowledge augmentation outperforms the baseline and partly the Whole Email run. This is also confirmed by the measures shown in Table 8.2.

Run	**MAP**	**P@5**	**P@10**	**P@15**	**P@20**	**P@30**
Baseline	0.2834	0.4915	0.4475	0.4102	0.3822	0.3333
knowlaug-mtarget-0.1	*0.2894*	0.5017	0.4542	0.4124	0.3822	*0.3418*
knowlaug-mtarget-0.2	*0.2958*	0.5119	0.4559	0.4124	*0.3915*	*0.3475*
knowlaug-mtarget-0.3	*0.3019*	0.5119	*0.4678*	0.4124	*0.3992*	*0.3508*
knowlaug-mtarget-0.4	*0.3051*	0.5186	0.4593	0.4260	*0.4068*	*0.3525*
knowlaug-mtarget-0.5	*0.3074*	0.5017	0.4525	*0.4373*	*0.4008*	*0.3503*
knowlaug-mtarget-0.6	*0.3106*	0.5085	0.4644	0.4339	0.3941	*0.3565*
knowlaug-mtarget-0.7	*0.3089*	0.4983	0.4729	0.4294	0.3975	*0.3548*
knowlaug-mtarget-0.8	*0.3066*	0.4814	0.4661	0.4328	0.3983	*0.3514*
knowlaug-mtarget-0.9	0.2992	0.4780	0.4508	0.4192	0.4000	0.3446
knowlaug-mtarget-1.0	0.2891	0.4780	0.4424	0.4090	0.3822	0.3339

Table 8.2: Mean average precision (MAP) and precision at K documents retrieved (P@K) of knowledge augmentation merged target runs

We can see that using the merged annotation targets as additional evidence and combining it by means of knowledge augmentation basically leads to better results. Many results show

Run	No merged target	One merged target
Baseline	5.30/1.78 (0.34)	4.70/2.62 (0.56)
knowlaug-mtarget-0.1	5.10/1.75 (0.34)	4.90/2.72 (0.55)
knowlaug-mtarget-0.2	4.82/1.67 (0.35)	5.18/2.82 (0.54)
knowlaug-mtarget-0.3	4.48/1.60 (0.36)	5.52/3.00 (0.54)
knowlaug-mtarget-0.4	4.12/1.47 (0.36)	5.88/3.05 (0.52)
knowlaug-mtarget-0.5	3.75/1.35 (0.36)	6.25/3.10 (0.50)
knowlaug-mtarget-0.6	3.37/1.25 (0.37)	6.63/3.32 (0.50)
knowlaug-mtarget-0.7	3.07/1.22 (0.40)	6.93/3.43 (0.50)
knowlaug-mtarget-0.8	2.60/1.05 (0.40)	7.40/3.53 (0.48)
knowlaug-mtarget-0.9	2.25/0.90 (0.40)	7.75/3.53 (0.46)
knowlaug-mtarget-1.0	1.92/0.77 (0.40)	8.08/3.60 (0.45)

Table 8.3: Average number of articles with merged targets, top 10 emails considered. The first number denotes the average number of emails, the second one the average number of emails that are relevant. The number in parentheses is the fraction of the second and first number.

an improvement in effectiveness compared to the baseline, and most of the improvements are statistically significant compared to the baseline. The mean average precision increases with increasing access probability, until it reaches its best value at access probability 0.6. Beyond this value, it decreases with further increasing access probability.

What is happening when we incorporate merged annotation targets as additional context for retrieval? With increasing access probability, the ratio of emails having no merged annotation targets to those having such obviously changes. This is confirmed by the numbers in Table 8.3, which are the basis for the further discussion. If we look, for example, at the top 10 emails in all the rankings, we find for the baseline on average 5.3 emails without merged targets in the ranking and 4.7 emails with. On average, 1.78 emails without merged targets are relevant (which gains a ratio of 0.34 of relevant vs. all emails with merged targets); with merged targets, 2.62 emails are relevant (ratio 0.56). The proportion of emails without merged targets to those with starts to change latest with an access probability of 0.2; now, we find more emails with merged targets than without in the top 10 emails (5.18 to 4.82). With 1.0 access probability, only 1.92 emails without merged targets are found in the top 10 emails on average. This tendency is also observable when looking at the top 5 or top 30 emails (the corresponding tables can be found in Appendix C.1.1). But how does it come that, at least until an access probability of 0.7 (if we look at P@10), this fluctuation in the ranking improves retrieval effectiveness? To get some insights, we look at the proportion of relevant documents to the total number of documents. For the baseline, we find that within the top 10 emails, 56% of the emails with merged targets are relevant on average. This ratio decreases with increasing access probability, which means that more irrelevant than relevant emails enter the top 10. This means that the emails entering the top 10 do not explain the increasing performance, as their quality gets worse. What seems to be interesting are the emails without merged targets which are replaced by those which. If we look at the emails without merged targets, we see that their quality increases. This means that proportionally more non-relevant emails without merged targets are kicked out of the top ranks and replaced by ones with merged targets, than relevant ones. So the potential of merged annotation targets (email quotations here) as a context lies in the fact that many non-relevant

documents can be replaced by those having merged targets, and more than half of them are relevant. The positive effect starts to vanish with higher access probabilities (> 0.7), when the ratio of emails with merged targets to those without gets higher, and the influence of emails without merged targets decreases. It seems that relevant emails without merged targets are basically able to "defend" their position in the top ranks.

From the experiments with merged annotation targets we can conclude that these constitute a valuable source which can be exploited in order to improve retrieval effectiveness. This underlines the importance of merged targets as a source to find out in which context an annotation was created (which is needed to understand its content). When we compare knowledge augmentation with merged targets to the Whole Email approach (which considers the same information), the results are mixed; knowledge augmentation does not seem to bring any significant improvement, but the opposite is true as well, so both approaches can be considered more or less equal. This means that modelling merged targets explicitly, as we do in POLAR, does not do any harm compared to the simpler approach of seeing the whole email as one document.

8.3.3.2 Fragments

Another focus of the experiments was the role of annotated fragments for discussion search. Annotated fragments are the parts of the email which were quoted and annotated again. The assumption is that these fragments are important parts of a message, and the terms contained in the fragments are good index terms. Knowledge augmentation with fragments does not introduce new terms, but raises the weights of the corresponding terms in an annotation or document.

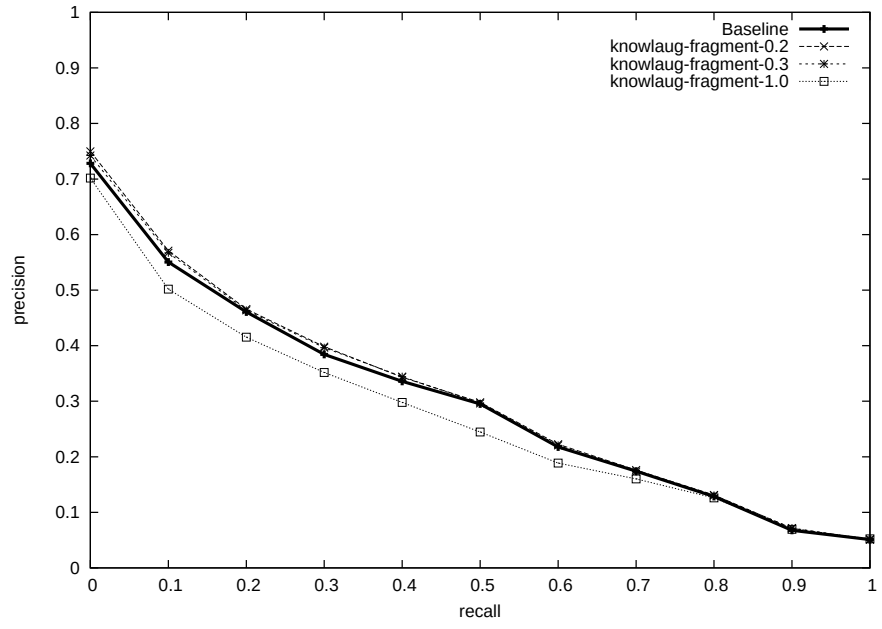


Figure 8.2: Recall-precision graph of selected knowledge augmentation fragment runs

Figure 8.2 shows the recall-precision-graph of selected fragment runs, again together with the baseline run for comparison. The gain in effectiveness is very big, but some fragment runs performed significantly better than the baseline. We can also see that some of the fragment

runs performed worse than the baseline. MAP and certain precision values are displayed in

Run	MAP	P@5	P@10	P@15	P@20	P@30
Baseline	0.2834	0.4915	0.4475	0.4102	0.3822	0.3333
knowlaug-fragment-0.1	*0.2883*	0.4983	0.4492	0.4124	*0.3898*	0.3367
knowlaug-fragment-0.2	*0.2908*	0.5153	0.4610	0.4124	0.3924	*0.3429*
knowlaug-fragment-0.3	0.2893	0.5186	0.4644	0.4147	0.3839	0.3356
knowlaug-fragment-0.4	0.2895	0.5085	0.4627	0.4102	0.3839	0.3316
knowlaug-fragment-0.5	0.2850	0.5017	0.4610	0.4079	0.3797	0.3271
knowlaug-fragment-0.6	0.2799	0.4983	0.4407	0.4090	0.3729	0.3237
knowlaug-fragment-0.7	0.2815	0.4915	0.4492	0.4034	0.3644	*0.3119*
knowlaug-fragment-0.8	0.2754	0.4881	0.4508	0.3864	0.3593	*0.3040*
knowlaug-fragment-0.9	0.2685	0.4881	0.4339	*0.3785*	*0.3475*	*0.2994*
knowlaug-fragment-1.0	0.2613	0.4847	0.4271	*0.3718*	*0.3407*	*0.2966*

Table 8.4: Mean average precision (MAP) and precision at K documents retrieved (P@K) of knowledge augmentation fragment runs

Table 8.4. Again, we see the tendency that with increasing access probability, MAP increases until the top is reached, and then decreases again. In this case, an access probability of 0.2 seems to mark the top, since the corresponding MAP value is still significantly higher compared to the baseline.

Obviously, an increasing access probability to fragments lets more emails with annotated fragments enter the top ranks (as we can see in Table 8.5 and in the tables in App. C.1.1). Along with this, we find that the percentage of relevant emails with fragments to all emails with fragments entering the top ranks basically decreases. One problem seems to be emails with more than 5 fragments. Their quality initially increases, but drops dramatically with an access probability higher than 0.2. Apparently there exist many non-relevant messages with quoted

	Number of fragments per article			
Run	0	>0	>5	>10
Baseline	7.12/2.62 (0.37)	2.88/1.78 (0.62)	0.00/0.00 (0.00)	0.00/0.00 (0.00)
knowlaug-fragment-0.1	6.83/2.52 (0.37)	3.17/1.90 (0.60)	0.03/0.02 (0.50)	0.00/0.00 (0.00)
knowlaug-fragment-0.2	6.30/2.40 (0.38)	3.70/2.13 (0.58)	0.12/0.07 (0.57)	0.00/0.00 (0.00)
knowlaug-fragment-0.3	5.73/2.30 (0.40)	4.27/2.27 (0.53)	0.20/0.08 (0.42)	0.02/0.00 (0.00)
knowlaug-fragment-0.4	5.23/2.15 (0.41)	4.77/2.42 (0.51)	0.22/0.10 (0.46)	0.02/0.00 (0.00)
knowlaug-fragment-0.5	4.80/2.05 (0.43)	5.20/2.48 (0.48)	0.28/0.10 (0.35)	0.05/0.00 (0.00)
knowlaug-fragment-0.6	4.15/1.72 (0.41)	5.85/2.62 (0.45)	0.35/0.10 (0.29)	0.05/0.00 (0.00)
knowlaug-fragment-0.7	3.53/1.58 (0.45)	6.47/2.83 (0.44)	0.33/0.10 (0.30)	0.05/0.00 (0.00)
knowlaug-fragment-0.8	3.10/1.47 (0.47)	6.90/2.97 (0.43)	0.35/0.10 (0.29)	0.05/0.00 (0.00)
knowlaug-fragment-0.9	2.72/1.33 (0.49)	7.28/2.93 (0.40)	0.37/0.10 (0.27)	0.05/0.00 (0.00)
knowlaug-fragment-1.0	2.38/1.23 (0.52)	7.62/2.97 (0.39)	0.37/0.10 (0.27)	0.05/0.00 (0.00)

Table 8.5: Average number of articles with fragments, top 10 emails considered. The first number denotes the average number of emails, the second one the average number of emails that are relevant. The number in parentheses is the fraction of the second and first number.

fragments containing query terms, and they get over-emphasised by higher access probabilities on the one hand and the amount of fragments on the other hand.

We observe an opposite tendency with emails not containing any annotated fragments: the quality of those remaining in the ranking increases (for top 10, 37% of the emails with no fragments are relevant in the baseline; with access probability 1.0, even 52% of the few remaining ones are relevant on average). Increasing the access probabilities kicks out many emails without fragments from the top positions, but these are mainly non-relevant ones, while relevant ones are often able to resist.

The bottom line is that also quoted fragments, as the parts of a message which are annotated, contain valuable contextual information which raises retrieval effectiveness when applied with knowledge augmentation. The knowledge about which fragments were annotated and thus important for the annotator helps to improve the ranking. Nevertheless, there is the danger that non-relevant documents having fragments with query terms are over-emphasised, especially when the number of fragments increases. This motivates the application of a more dynamic access probability which considers the number of fragments for future experiments.

8.3.3.3 Content Annotations

Another series of experiments dealt with the role of content annotations in the knowledge augmentation process for discussion search. Only direct annotations were considered (radius-1 augmentation). If terms appear both in the unquoted body of an email and in the new part of its reply, then the certainty that we could index the email with the common term was raised. This way, we hope to gain a better precision. On the other hand, direct annotations might introduce new terms associated with the topic of the annotated object, which could result in a better recall.

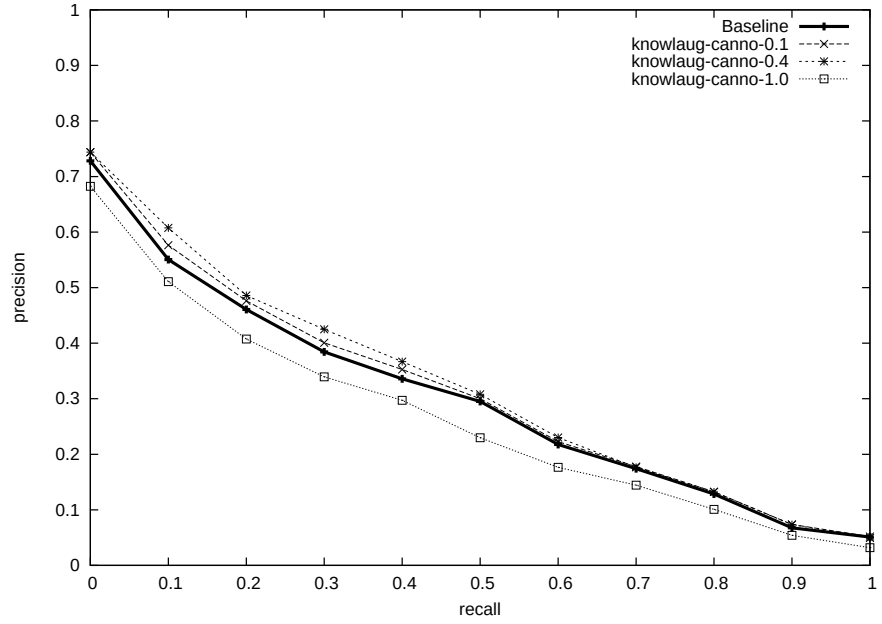


Figure 8.3: Recall-precision graph of selected knowledge augmentation content annotation runs

Figure 8.3 shows the recall-precision graph of selected knowledge augmentation content annotation runs, together with the baseline run for comparison. Many content annotation runs significantly outperform the baseline, although with a high access probability, results tend to become significantly worse than the baseline. Table 8.6 shows the MAP and certain precision values. We observe a similar tendency as with merged targets and fragments. MAP raises until an access probability of 0.4, and then decreases again. For P@5, P@10, we gain the best results with an access probability of 0.3. The results for these values are all statistically significant.

Run	**MAP**	**P@5**	**P@10**	**P@15**	**P@20**	**P@30**
Baseline	0.2834	0.4915	0.4475	0.4102	0.3822	0.3333
knowlaug-canno-0.1	*0.2942*	*0.5220*	*0.4695*	0.4181	*0.3941*	*0.3435*
knowlaug-canno-0.2	*0.3003*	*0.5288*	*0.4949*	0.4249	*0.4000*	*0.3503*
knowlaug-canno-0.3	*0.3035*	*0.5390*	*0.5034*	*0.4441*	*0.4000*	*0.3508*
knowlaug-canno-0.4	*0.3042*	0.5356	*0.5034*	*0.4452*	*0.4042*	*0.3508*
knowlaug-canno-0.5	*0.3019*	0.5119	*0.4932*	*0.4475*	0.4059	0.3492
knowlaug-canno-0.6	0.2948	0.5051	*0.4949*	*0.4429*	0.4017	0.3503
knowlaug-canno-0.7	0.2842	0.5220	0.4729	0.4305	0.4017	0.3486
knowlaug-canno-0.8	0.2727	0.5051	0.4542	0.4203	0.3958	0.3345
knowlaug-canno-0.9	*0.2596*	0.4814	0.4407	0.4079	0.3814	0.3237
knowlaug-canno-1.0	*0.2481*	0.4610	0.4305	0.3921	0.3661	0.3124

Table 8.6: Mean average precision (MAP) and precision at K documents retrieved (P@K) of knowledge augmentation content annotation runs

Table 8.7 shows the average number of articles with content annotations in the top 10 ranks (further numbers for top 5 and top 30 can be found in Appendix C.1.1). Again, we observe that the quality of the emails without content annotations increases with increasing access probabilities, while their total number decreases. The ratio of relevant emails having content

	Number of content annotations per article			
Run	0	>0	>5	>10
Baseline	6.22/2.30 (0.37)	3.78/2.10 (0.56)	0.00/0.00 (0.00)	0.00/0.00 (0.00)
knowlaug-canno-0.1	5.68/2.17 (0.38)	4.32/2.45 (0.57)	0.05/0.03 (0.67)	0.00/0.00 (0.00)
knowlaug-canno-0.2	5.08/2.02 (0.40)	4.92/2.85 (0.58)	0.15/0.10 (0.67)	0.00/0.00 (0.00)
knowlaug-canno-0.3	4.58/1.90 (0.41)	5.42/3.05 (0.56)	0.28/0.10 (0.35)	0.03/0.00 (0.00)
knowlaug-canno-0.4	4.05/1.72 (0.42)	5.95/3.23 (0.54)	0.43/0.15 (0.35)	0.07/0.00 (0.00)
knowlaug-canno-0.5	3.52/1.55 (0.44)	6.48/3.30 (0.51)	0.50/0.15 (0.30)	0.07/0.00 (0.00)
knowlaug-canno-0.6	2.95/1.40 (0.47)	7.05/3.47 (0.49)	0.57/0.13 (0.24)	0.07/0.00 (0.00)
knowlaug-canno-0.7	2.48/1.17 (0.47)	7.52/3.48 (0.46)	0.60/0.13 (0.22)	0.05/0.00 (0.00)
knowlaug-canno-0.8	2.00/0.93 (0.47)	8.00/3.53 (0.44)	0.62/0.13 (0.22)	0.05/0.00 (0.00)
knowlaug-canno-0.9	1.67/0.87 (0.52)	8.33/3.47 (0.42)	0.60/0.12 (0.19)	0.05/0.00 (0.00)
knowlaug-canno-1.0	1.33/0.80 (0.60)	8.67/3.43 (0.40)	0.63/0.12 (0.18)	0.03/0.00 (0.00)

Table 8.7: Average number of articles with content annotations, top 10 emails considered. The first number denotes the average number of emails, the second one the average number of emails that are relevant. The number in parentheses is the fraction of the second and first number.

annotations to emails with content annotations in the top 10 ranks increases until an access probability of 0.3, when it starts to decrease. This explains why access probabilities around 0.3 and 0.4 gain the best results, but results get worse with higher access probabilities and an increasing dominance of emails with content annotations.

The experiments show that for discussion search, the direct annotations are an important and strong context to consider. Direct annotations can indeed help to improve the precision of the returned rankings. With increasing access probability, the bias from direct annotations becomes too large.

8.3.3.4 All Evidence

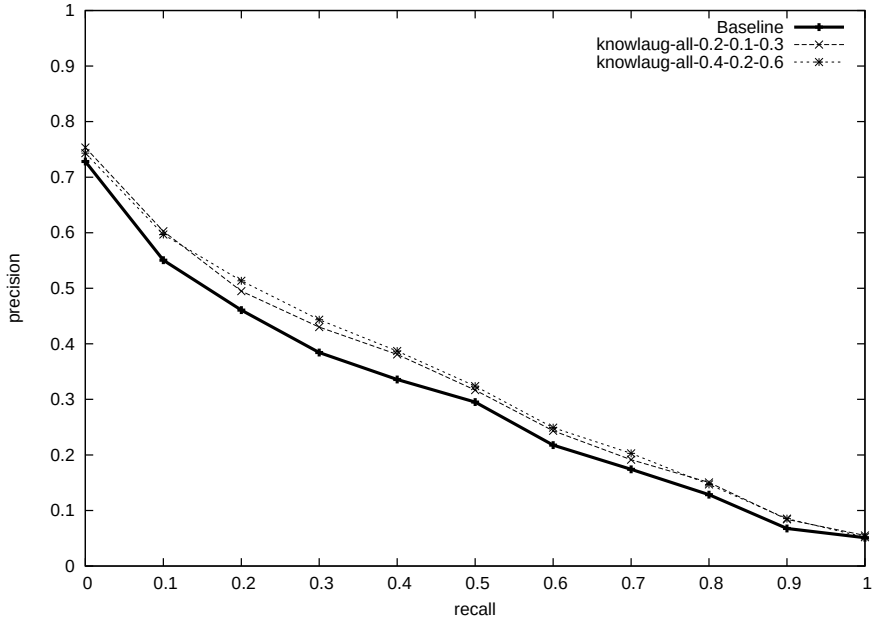


Figure 8.4: Recall-precision graph of selected knowledge augmentation all evidence runs

The final discussion search experiments combined the evidence coming from merged targets, fragments and direct content annotations. Figure 8.4 shows the recall-precision curves of selected runs, Table 8.8 the precision values of the performed runs. The runs knowlaug-all-0.2-0.1-0.3 and knowlaug-anno-0.4-0.2-0.6 performed significantly better than the baseline. Both experiments produced the best performance values observed[4]. The combined runs even outperform the Whole Email run, although only the P@15 values are statistically significant compared to that run.

We showed in the experiments that combining all evidence can basically generate the best results. This confirms that using the rich structure made of merged targets, fragments and content annotations can improve retrieval effectiveness.

[4]Note that these experiments are two examples of combined runs. Not all possible combinations were tried, so there may be combinations which perform even better than the ones presented here.

Run	MAP	P@5	P@10	P@15	P@20	P@30
Baseline	0.2599	0.4441	0.4102	0.3955	0.3695	0.3220
knowlaug-all-0.2-0.1-0.3	*0.3139*	***0.5424***	*0.5034*	*0.4452*	*0.4068*	*0.3616*
knowlaug-all-0.4-0.2-0.6	***0.3205***	0.5288	***0.5068***	***0.4633***	***0.4161***	***0.3689***

Table 8.8: Mean average precision (MAP) and precision at K documents retrieved (P@K) of knowledge augmentation all evidence runs

8.4 Document Search

The document search experiments, which are also reported in Frommholz (2007), targeted the following questions:

> Can knowledge augmentation, where the content of an article is augmented with the content of the connected discussion threads, enhance retrieval effectiveness of annotation-based document search? Furthermore, do we need to consider all comments in the discussion threads (*full augmentation*) or is it sufficient to consider only the direct comments for knowledge augmentation (*radius-1 augmentation*)?

The ZDNet News collection introduced in Section 7.2 was used for the document search experiments.

8.4.1 Description of Runs

The following runs with specific parameter settings were performed (consider the POLAR representation of ZDNet articles and comments in Section 7.2.3):

Baseline In the baseline run, we only looked at the articles themselves, but did not perform any knowledge augmentation. For example, the article zdnet_art1 in Section 7.2.3 and the query seeking for relevant articles would then be represented as

```
zdnet_art1[ 0.7 sun   0.6 open-source   0.8 solaris   0.4 era
            0.4 *zdnet_com1   0.4 *zdnet_com2 ...
          ]
?- D->q & document(D)
```

Merged In this run, articles and their direct annotations are merged and viewed as a single document. This is the approach used to create an initial ranking for relevance judgements (see Section 7.2.2). In POLAR, this is represented as

```
zdnet_art1[ 0.9 sun   0.95 open-source  0.85 solaris   0.5 era
            0.4 unix
          ]
?- D->q & document(D)
```

Note that in this representation, the terms from zdnet_com1 and zdnet_com2 are merged with the original terms from zdnet_art1, and also the document length changed; therefore the different term weights.

knowlaug-<acc_prob> In these runs, knowledge augmentation is performed which considers all discussion threads attached to an article. <acc_prob> denotes the corresponding

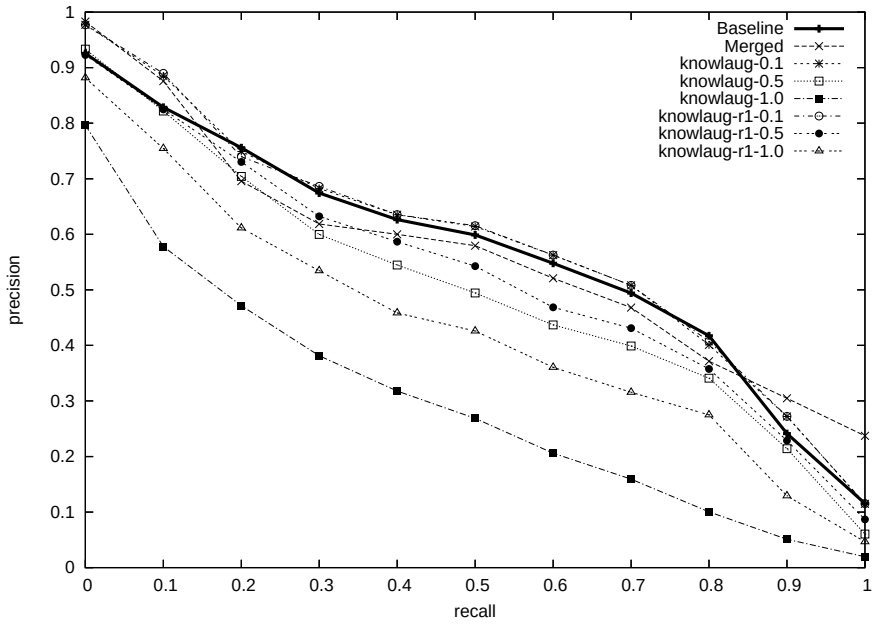


Figure 8.5: Recall-precision graph of selected document search runs. Relevant documents are the ones judged so.

access probability; for instance, the run "knowlaug-0.1" means knowledge augmentation with access probability 0.1. Then we get for zdnet_art1 and the query:

```
zdnet_art1[ 0.7 sun   0.6 open-source   0.8 solaris   0.4 era
            0.1 *zdnet_com1   0.1 *zdnet_com2 ...
          ]
?- //D->q & document(D)
```

knowlaug-r1-<acc_prob> In these runs, knowledge augmentation is performed which considers only direct comments attached to an article. <acc_prob> denotes the corresponding access probability. The POLAR representation is the same as the one for knowlaug-<acc_prob>, but a radius-1 augmentation algorithm is applied instead.

The runs are evaluated with respect to the two different relevance criteria (see Section 7.2.2):

1. Documents are relevant only if they were judged so;
2. Documents are relevant if they were judge so or have relevant direct comments.

The second experiment might be interesting for users which like to consider relevant documents but would also use them as a starting point to browse the discussions for relevant information.

8.4.2 Results

Figure 8.4.2 shows the results of some selected runs where documents are relevant only if judged so. Table 8.9 presents some precision values for all run. We can see that the bias coming from the annotations should not be too strong. The retrieval quality decreases with increasing

access probability. But we also see that a slight bias, when the access probability is 0.1, seems to be beneficial (although the results are not statistically significant). We also discover that the difference between performing full knowledge augmentation vs. radius-1 knowledge augmentation is only marginal, so it seems fine to apply radius-1 knowledge augmentation instead of traversing whole annotation threads. In fact, as we can see in the recall-precision graph in Figure 8.4.2, for high access probabilities full knowledge augmentation has a more destructive effect. This can be explained by topic changes occurring in a discussion thread. If the terms describing a new topic after a topic change are propagated with a high access probability to the root document, the algorithm assumes this document to be relevant to the new topic, which it is most probably not. With low access probability, this effect vanishes, and with radius-1 knowledge augmentation, the probability of a topic change is small as we regard only direct annotations here.

Run	MAP	P@5	P@10	P@15	P@20	P@30
Baseline	0.5609	0.7700	0.7000	0.6600	0.6150	0.5467
Merged	0.5511	0.7700	0.6600	0.5900	0.5625	0.5100
knowlaug-0.1	**0.5773**	**0.7800**	0.7050	**0.6867**	**0.6300**	**0.5517**
knowlaug-0.2	0.5627	0.7400	0.7050	0.6700	0.6200	0.5383
knowlaug-0.3	0.5454	0.7300	**0.7100**	0.6533	0.6150	*0.5200*
knowlaug-0.4	*0.5233*	0.7400	0.6750	0.6400	0.5825	*0.5017*
knowlaug-0.5	*0.4945*	0.7100	0.6650	*0.6000*	*0.5425*	*0.4783*
knowlaug-0.6	*0.4633*	0.6800	0.6450	*0.5600*	*0.5150*	*0.4517*
knowlaug-0.7	*0.4268*	*0.6500*	*0.5900*	*0.5367*	*0.4925*	*0.4317*
knowlaug-0.8	*0.3872*	*0.6100*	*0.5500*	*0.5100*	*0.4650*	*0.3950*
knowlaug-0.9	*0.3462*	*0.6000*	*0.4850*	*0.4833*	*0.4275*	*0.3583*
knowlaug-1.0	*0.2792*	*0.5100*	*0.4550*	*0.4000*	*0.3575*	*0.3267*
knowlaug-r1-0.1	0.5768	**0.7800**	**0.7100**	**0.6867**	0.6275	**0.5517**
knowlaug-r1-0.2	0.5670	0.7500	0.7050	0.6733	0.6275	0.5417
knowlaug-r1-0.3	0.5514	0.7100	0.7000	0.6700	0.6225	0.5300
knowlaug-r1-0.4	0.5327	*0.6800*	0.6800	0.6433	0.6075	*0.5100*
knowlaug-r1-0.5	*0.5143*	*0.6700*	0.6550	0.6200	*0.5725*	*0.5067*
knowlaug-r1-0.6	*0.4980*	0.6800	0.6500	*0.6100*	*0.5525*	*0.4967*
knowlaug-r1-0.7	*0.4789*	*0.6700*	0.6550	*0.5867*	*0.5300*	*0.4800*
knowlaug-r1-0.8	*0.4579*	*0.6400*	0.6400	*0.5767*	*0.5200*	*0.4517*
knowlaug-r1-0.9	*0.4365*	*0.6000*	*0.6250*	*0.5633*	*0.5075*	*0.4367*
knowlaug-r1-1.0	*0.4184*	*0.6000*	*0.6000*	*0.5433*	*0.4950*	*0.4250*

Table 8.9: Mean average precision (MAP) and precision at K documents retrieved (P@K) for the document search runs. Relevant documents are the ones judged so.

The following considerations are based on Table 8.10; additional tables can be found in Appendix C.1.2. As expected, most articles in the top ranks of the baseline run have comments (for top 10, we find on average 1.7 articles without comments and 8.3 with), since most articles in the whole collection are commented. With increasing access probability, the number of articles with comments increases, while the number of articles without decreases (top 10 and access probability 1.0: 9.87 articles on average with at least one comment, only 0.13 without

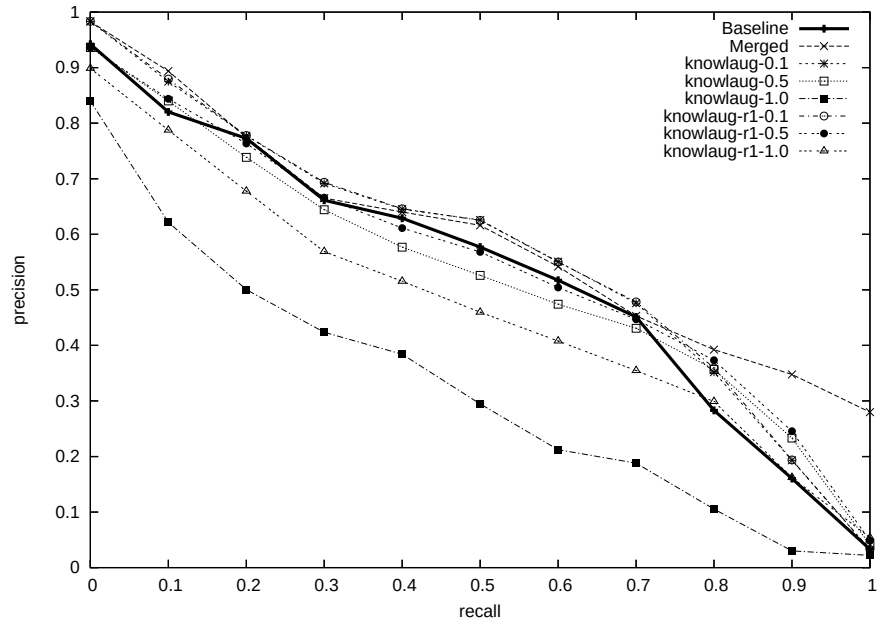


Figure 8.6: Recall-precision graph of selected document search runs. Relevant documents are the ones judged so or having relevant annotations.

(full augmentation)). Looking at the articles with comments in the top ranks, we find that the proportion of relevant articles among them decreases with increasing access probability (from 66% in the baseline to 41% with access probability 1.0, top 10 with full augmentation). With radius-1 augmentation, this proportion decreases less (only down to 54% for top 10). On the other hand, for articles without annotations, the proportion of relevant ones increases, and even goes up to 1.0 for high access probabilities in the top 5 and top 10 ranks (probably due to the fact that in these cases, articles without comments barely appear in the top ranks). We again have two opposing tendencies which we already observed in the discussion search experiments: quality decreases for articles with comments, and increases for articles without. This time, due to the huge ratio of articles with comments on articles without, the former effect dominates, which leads to worse results with increasing access probabilities. Similar as for fragments in discussion search, we see the tendency that articles with many comments tend to enter the top ranks due to their amount of comments, and proportionally more of them are non-relevant compared to articles with less comments. Like for fragments, this again motivates the usage of an access probability which also depends on the number of comments. One reason why we observe a slight improvement only with a small access probability seems to be that a global access probability is not feasible here.

A weaker relevance criterion used for the relevance judgements described in Section 7.2.2 was to regard an article as relevant if it is judged so itself or if it contains at least one relevant annotation. This was applied for the results in Table 8.11. We can see a bigger gain in retrieval effectiveness w.r.t. the baseline for access probabilities 0.1 and 0.2, and the Merged run. Some results are even statistically significant. This is of course not a big surprise, as our baseline run does not consider the additional knowledge coming from (relevant) annotations. We also observe the tendency to worse results with increasing access probabilities (Fig. 8.6).

Run	Number of direct comments per article						
	0	>0	>5	>10	>50	>100	>200
Baseline	1.70/0.91 (0.54)	8.30/5.48 (0.66)	4.91/3.17 (0.65)	3.04/1.91 (0.63)	0.35/0.26 (0.75)	0.09/0.04 (0.50)	0.00/0.00 (0.00)
knowlaug-0.1	1.26/0.74 (0.59)	8.74/5.70 (0.65)	6.04/3.83 (0.63)	4.30/2.70 (0.63)	1.13/0.83 (0.73)	0.30/0.22 (0.71)	0.13/0.09 (0.67)
knowlaug-0.2	1.00/0.61 (0.61)	9.00/5.78 (0.64)	6.83/4.22 (0.62)	5.17/3.17 (0.61)	1.78/1.04 (0.59)	0.70/0.35 (0.50)	0.30/0.09 (0.29)
knowlaug-0.3	0.74/0.48 (0.65)	9.26/5.96 (0.64)	7.22/4.39 (0.61)	5.57/3.30 (0.59)	2.00/1.04 (0.52)	0.78/0.35 (0.44)	0.35/0.09 (0.25)
knowlaug-0.4	0.65/0.39 (0.60)	9.35/5.70 (0.61)	7.48/4.30 (0.58)	5.78/3.17 (0.55)	2.52/1.09 (0.43)	1.22/0.35 (0.29)	0.43/0.09 (0.20)
knowlaug-0.5	0.48/0.35 (0.73)	9.52/5.65 (0.59)	7.61/4.30 (0.57)	5.87/3.13 (0.53)	2.52/1.04 (0.41)	1.26/0.30 (0.24)	0.43/0.09 (0.20)
knowlaug-0.6	0.39/0.30 (0.78)	9.61/5.57 (0.58)	7.61/4.22 (0.55)	5.91/3.04 (0.51)	2.70/1.04 (0.39)	1.35/0.30 (0.23)	0.48/0.09 (0.18)
knowlaug-0.7	0.30/0.26 (0.86)	9.70/5.13 (0.53)	8.04/4.00 (0.50)	6.43/3.00 (0.47)	3.13/1.04 (0.33)	1.52/0.30 (0.20)	0.52/0.09 (0.17)
knowlaug-0.8	0.17/0.17 (1.00)	9.83/4.83 (0.49)	8.30/3.83 (0.46)	6.78/2.91 (0.43)	3.26/1.00 (0.31)	1.52/0.26 (0.17)	0.48/0.09 (0.18)
knowlaug-0.9	0.17/0.17 (1.00)	9.83/4.26 (0.43)	8.61/3.57 (0.41)	7.26/2.83 (0.39)	3.43/0.87 (0.25)	1.57/0.22 (0.14)	0.48/0.09 (0.18)
knowlaug-1.0	0.13/0.13 (1.00)	9.87/4.04 (0.41)	8.74/3.43 (0.39)	7.43/2.70 (0.36)	3.09/0.70 (0.23)	1.43/0.22 (0.15)	0.43/0.09 (0.20)
knowlaug-r1-0.1	1.26/0.74 (0.59)	8.74/5.74 (0.66)	6.04/3.87 (0.64)	4.26/2.70 (0.63)	1.13/0.83 (0.73)	0.30/0.22 (0.71)	0.13/0.09 (0.67)
knowlaug-r1-0.2	1.04/0.61 (0.58)	8.96/5.83 (0.65)	6.74/4.22 (0.63)	5.04/3.09 (0.61)	1.70/1.00 (0.59)	0.70/0.35 (0.50)	0.30/0.09 (0.29)
knowlaug-r1-0.3	0.83/0.52 (0.63)	9.17/5.83 (0.64)	7.04/4.26 (0.60)	5.43/3.22 (0.59)	2.00/1.09 (0.54)	0.78/0.35 (0.44)	0.35/0.09 (0.25)
knowlaug-r1-0.4	0.70/0.43 (0.62)	9.30/5.70 (0.61)	7.39/4.26 (0.58)	5.70/3.13 (0.55)	2.30/1.04 (0.45)	1.00/0.35 (0.35)	0.43/0.09 (0.20)
knowlaug-r1-0.5	0.65/0.39 (0.60)	9.35/5.52 (0.59)	7.43/4.17 (0.56)	5.74/3.04 (0.53)	2.52/1.04 (0.41)	1.22/0.35 (0.29)	0.48/0.09 (0.18)
knowlaug-r1-0.6	0.52/0.35 (0.67)	9.48/5.52 (0.58)	7.52/4.13 (0.55)	5.78/3.00 (0.52)	2.65/1.09 (0.41)	1.30/0.35 (0.27)	0.48/0.09 (0.18)
knowlaug-r1-0.7	0.43/0.35 (0.80)	9.57/5.57 (0.58)	7.57/4.17 (0.55)	5.83/3.04 (0.52)	2.65/1.04 (0.39)	1.22/0.30 (0.25)	0.39/0.09 (0.22)
knowlaug-r1-0.8	0.35/0.30 (0.88)	9.65/5.48 (0.57)	7.78/4.17 (0.54)	5.96/3.00 (0.50)	2.78/1.04 (0.38)	1.39/0.30 (0.22)	0.39/0.09 (0.22)
knowlaug-r1-0.9	0.35/0.30 (0.88)	9.65/5.35 (0.55)	7.78/4.13 (0.53)	6.09/3.00 (0.49)	2.87/1.04 (0.36)	1.39/0.30 (0.22)	0.39/0.09 (0.22)
knowlaug-r1-1.0	0.17/0.17 (1.00)	9.83/5.26 (0.54)	7.83/4.00 (0.51)	6.17/2.91 (0.47)	2.96/1.04 (0.35)	1.39/0.30 (0.22)	0.39/0.09 (0.22)

Table 8.10: Average number of articles with content annotations, top 10 emails considered. The first number denotes the average number of emails, the second one the average number of emails that are relevant. The number in parentheses is the fraction of the second and first number.

Run	MAP	P@5	P@10	P@15	P@20	P@30
Baseline	0.5257	0.7800	0.7100	0.6667	0.6225	0.5550
Merged	**0.5828**	**0.8100**	0.7100	0.6400	0.6075	0.5600
knowlaug-0.1	*0.5605*	**0.8100**	0.7200	*0.7000*	0.6450	0.5667
knowlaug-0.2	*0.5596*	0.7700	0.7250	0.6900	0.6375	0.5650
knowlaug-0.3	0.5541	0.7600	**0.7350**	0.6833	0.6400	0.5533
knowlaug-0.4	0.5407	0.7700	0.7100	0.6767	0.6175	0.5417
knowlaug-0.5	0.5183	0.7500	0.7000	0.6433	0.5800	*0.5217*
knowlaug-0.6	0.4916	0.7200	0.6850	*0.6033*	*0.5625*	*0.5017*
knowlaug-0.7	*0.4559*	0.6900	*0.6350*	*0.5867*	*0.5425*	*0.4850*
knowlaug-0.8	*0.4178*	*0.6700*	*0.5950*	*0.5600*	*0.5200*	*0.4467*
knowlaug-0.9	*0.3758*	*0.6500*	*0.5450*	*0.5300*	*0.4825*	*0.4083*
knowlaug-1.0	*0.3091*	*0.5700*	*0.5200*	*0.4600*	*0.4100*	*0.3733*
knowlaug-r1-0.1	*0.5595*	**0.8100**	0.7250	***0.7033***	0.6425	***0.5683***
knowlaug-r1-0.2	*0.5616*	0.7800	0.7250	0.6900	0.6450	**0.5683**
knowlaug-r1-0.3	*0.5569*	0.7400	0.7200	0.6967	**0.6475**	0.5600
knowlaug-r1-0.4	0.5463	0.7200	0.7150	0.6767	0.6350	0.5450
knowlaug-r1-0.5	0.5349	0.7000	0.6900	0.6567	0.6075	0.5483
knowlaug-r1-0.6	0.5235	0.7200	0.6850	0.6533	0.5950	0.5433
knowlaug-r1-0.7	0.5074	*0.6900*	0.7000	0.6367	*0.5750*	0.5300
knowlaug-r1-0.8	0.4897	*0.6800*	0.6850	0.6300	*0.5675*	*0.5017*
knowlaug-r1-0.9	*0.4709*	*0.6400*	0.6700	0.6200	*0.5575*	*0.4867*
knowlaug-r1-1.0	*0.4543*	*0.6500*	0.6550	*0.5933*	*0.5475*	*0.4767*

Table 8.11: Mean average precision (MAP) and precision at K documents retrieved (P@K) for the document search runs. Relevant documents are the ones judged so or having relevant annotations.

What difference it makes to apply the weaker relevance criterion, compared to the stricter one (where articles are relevant only if judged so), can be observed in one topic about "Firefox security". Here, the difference between the Baseline MAP and the knowlaug-0.1 MAP is 0.09 in the strict case and -0.14 in the weak one, so knowledge augmentation performed much better in the weak and worse in the strict case. In this particular topic, many non-relevant articles are judged as having relevant annotations. For example, in an article about Microsoft's Internet Explorer (IE) being divorced from Windows, also some Firefox security issues are mentioned in the annotations (in fact, in this particular article, many discussions arose about Firefox vs. IE in general, which led this article to be ranked 2nd place for the query about Firefox security). Topics like this, having many documents with relevant annotations, thus benefit from our knowledge augmentation as well as from the merged approach. It is up to the actual application if such documents, which are themselves not relevant, but contain interesting information for the user in their annotations, should be retrieved. It seems to make sense, because otherwise relevant information would have been missed.

8.5 Determining the Polarity of an Annotation

Some further experiments were performed to automatically classify annotations with respect to their *polarity*. Such a classification is for instance used in the trustworthiness showcase in Section 4.3.8 on page 79 and provides an important non-topical measure for certain information needs. While in systems like COLLATE (see Section 2.1.2.2) we can infer from the annotation type that an annotation is positive or negative (for example, the *counterargument* type would be negative, whereas *support argument* or *elaboration* would be positive), in most systems there is no explicit evidence about the polarity of annotations. This information might either be provided manually, which is a very tedious task, or automatically, for example by machine learning. The latter one is the approach we are following here. We briefly describe the results of some experiments which are reported more thoroughly by Lechtenfeld (2007) and also in Frommholz and Lechtenfeld (2008).

8.5.1 Machine Learning for Sentiment Classification in Discussions

The idea is to adapt machine learning approaches known from sentiment classification, where the goal is to determine if messages like blog entries talk positively or negatively about a specific product (see, e.g., Pang et al. (2002)), to the classification of ZDNet comments into positive and negative ones. For this, the test collection described in Section 7.2.4 (p. 190) was used. From this test collection, the three classes positive, negative and neutral were derived (so there was no distinction between content and meta annotations).

Support Vector Machines (SVM) (Joachims, 1998) were used for the classification task. The main challenge of the task at hand is to find suitable features representing the sentiment of a comment. The SVM had to be trained and these features had to be extracted. The features can be classified as follows:

- *textual features* like term occurrences (unigrams or bigrams), location (title, previous comment, first term of comment), negation, comment length;
- *references, mentioning of authors, article topics* – are other pages referenced, is the author of the previous comment mentioned in the reply, what is the topic of the article the thread belongs to;
- *context features regarding authors' response behaviour*, comprising the response time of replies, duration of replying, day of week and hour of reply, number of replies to current and previous comment, number of replies in subthread starting with current comment;
- *thread and comment structure*: is the current title the same as the previous one, is the comment a direct reply to another comment or to the corresponding article, sequence of authors, amount of quotations.

A combination of above features on a binary SVM (which directly classifies a comment into positive or negative) achieved an accuracy[5] of 0.79. Further experiments were performed using a binary metaclassifier. Such a metaclassifier consists of three basic SVMs which classify an annotation into positive/negative, positive/neutral and neutral/negative, respectively. The output of the three classifiers is then used by the metaclassifier to determine the final category[6].

[5] Number of correctly classified instances divided by the total number of classified instances

[6] A metaclassifier is usually applied when SVMs should make a classification decision comprising more than two classes, but it can also be used to refine binary classification decisions.

Surprisingly, neglecting the textual features gained the overall best accuracy of 0.8 for the binary multiclassifier.

The main conclusion from the sentiment classification experiments is that this is more difficult task than a text categorisation task, but the results look surprisingly promising, which also makes the inclusion of measures like the trustworthiness into the retrieval function realistic. The present experiments were meant to gain a feeling of how well a classification into positive and negative comments could perform. Future work might refine this approach or develop new ones to improve these first results.

8.6 Summary and Discussion

In this chapter, POLAR's core retrieval functionality, knowledge augmentation with probabilistic inference estimating $P(d \rightarrow q)$, was evaluated. We presented experiments with discussion search (using W3C lists) and document search (using ZDNet News). Additionally, we outlined the results of experiments to determine the polarity of annotations, which is important for approaches satisfying sophisticated information needs, e.g. by measuring the trustworthiness of annotations based on the number of positive and negative replies.

For the discussion search experiments, we applied the annotation view on email messages explained in Section 7.1.2, which gave us a full set of fragments, merged targets and content annotations. The goal of the evaluation was therefore to show if annotated fragments, merged annotation targets and content annotations can improve retrieval effectiveness when used as a context for discussion search. The results showed a significant improvement over a baseline where only the new parts of emails were considered. The best results were reported for a combination of merged annotation targets, fragments and content annotations. The annotation view on email messages and its representation in POLAR is thus a good choice for email-based discussion search, with the additional advantage that it supports structure queries to the knowledge base.

The document search experiments, where comments on ZDNet News articles were used as a context, revealed that only a very low access probability leads to minor improvements in retrieval effectiveness. The higher the access probability, the worse the results (which then go below the baseline). It seems to be sufficient to use radius-1 augmentation instead of the (more expensive) full augmentation, since otherwise topic changes in the discussion seem to have a negative influence. Such topic changes also occur within radius-1 annotations, which leads to a slightly better performance when regarding documents as relevant if they are judged so or have relevant direct comments.

The machine learning approach to determine the polarity of an annotation uses Support Vector Machines and applies textual features as well as references, mentioning of authors, article topics, context features regarding authors' response behaviour, and features coming from the thread and comment structure. The polarity of annotations can then be determined with an accuracy of about 0.8.

One of the conclusions to draw from the discussion search experiments is that if we model annotation-based discussion similar to email, that is we can identify (merged) annotation targets as pieces of the annotated object, and therefore also the fragments which are annotated, this rich structure is a good context for annotation-based discussion search applying POLAR's knowledge augmentation facilities. Since annotation authoring systems like the Multivalent Browser (Phelps and Wilensky, 1997) contain information about annotated fragments, there

is a good chance that retrieval tools like POLAR are able to exploit this rich context given by annotation targets, annotated fragments and content annotations. And of course we have shown that POLAR is a tool for email discussion search when applying the annotation view on emails. The results of the experiments emphasise the usefulness of the model discussed in Chapter 3 and confirm the decision to introduce merged targets and fragments as subcontexts in POLAR.

In all experiments, we observe a common pattern: there must be a good balance between access probabilities and the number of subcontexts. For example, per email there can only be one merged target in the W3C lists collection; we saw here that a relatively high access probability could achieve good results. On the other hand, articles in ZDNet News tend to have many annotations, so only a low access probability (around 0.1) leads to at least slight improvements. Generally, in collections like ZDNet News and other sites where users can annotate articles, popular articles have many annotations, most probably more than comments have replies. This means that for document search, a lower access probability should be applied than for discussion search, where there are not so many subcontexts to consider for augmentation.

Regarding the document search experiments, we should be aware that the ZDNet testbed with 20 topics and 150 documents judged per topic, and the fact that some results are not statistically significant has an effect on the reliability of our results (Sanderson and Zobel, 2005). However, we conclude that in essence there is an improvement in retrieval effectiveness for the given settings by applying knowledge augmentation in a very moderate way (global access probabilities around 0.1 and 0.2), especially in cases where it is sufficient that non-relevant articles have relevant annotations (at least some of the results are significant here).

The bottom line is that POLAR's knowledge augmentation approach is basically able to improve retrieval effectiveness for both discussion and document search. We can conclude that, as initially assumed, annotations and their "indirect" objects like fragments and annotation targets contain information from which the retrieval process can benefit when using them as a context. This is certainly a good result for the field of annotation-based retrieval. But we have to keep in mind that both the W3C lists and the ZDNet collection contain a very special kind of annotation, as discussed in the previous chapter. W3C lists mainly consist of experts' discussion, whereas comments in ZDNet sometimes lack substance regarding their quality. While we find such annotations in various news portals on the web, we cannot necessarily expect that our results are valid for other kinds of annotations (like personal notes or annotations in scholarly environments or humanities) or subject areas, as the type and quality of annotations might differ. Further experiments certainly need to be performed to learn if and how the types and subject areas of annotations affect results, and crucial for this is the creation of new test collections, as the ones applied here only cover a narrow set of annotations. Furthermore, it would be desirable to perform experiments with meta annotations and distinct annotation types (like questions, comments, etc).

The polarity experiments with support vector machines to distinguish between positive and negative comments led to an accuracy of around 0.79 and 0.8. This shows that this task is more challenging than the well-understood text categorisation task, but the results are promising. A possible next step would be to integrate the results of such a sentiment classification in discussions to a retrieval framework which includes a measure of the trustworthiness of annotations, as it is reported in Section 4.3.8. A suitable test collection which also considers the trustworthiness as a relevance criterion besides topical relevance needs to be created; once such a collection is available, the effect of the categorisation accuracy on the retrieval effectiveness can be measured.

9

Conclusion and Outlook

> It's been a long road, getting from there to here. It's been a long time, but my time is fin'lly near.
>
> *(From the opening theme of "Star Trek: Enterprise")*

Many applications allow users to annotate the material at hand. Annotations can be comments, markings (highlighted parts) and references to other objects. Typically, users select the object they want to annotate first; such an object can be a whole document, a set of documents, but also a document part or only a fragment. When dealing with collections containing documents and annotations, important information a user seeks might be contained in the annotations, but not in the main documents. Annotations thus establish an important context documents and even the annotations themselves are embedded in. The challenge is to exploit the annotation context to uncover the information contained in annotations on the one hand, and to determine the relevance of main documents w.r.t. the user's query on the other hand. This thesis therefore tried to answer the following question:

> How and how effective can we exploit annotations for information retrieval?

To answer this question, the following three sub questions were formulated:

1. What are annotations, and what is the context established by annotations?
2. How can we model the annotation context, query it and use it for annotation-based retrieval to satisfy advanced information needs?
3. How effective are methods exploiting the annotation context; do annotations help to gain a better retrieval performance?

These questions were answered in the three parts of this thesis.

Part I dealt with the question what the annotation context actually is, which objects play a role there and how they are related. Nowadays, annotations can be found in many applications on the Web (be it the "normal" Web, the so-called "Web 2.0" or the Semantic Web) or in digital libraries, and they serve different purposes. Annotations may be comments, reviews, judgements, but also references which relate objects in a repository where no such relation existed before. Annotations can be the building block for establishing scientific discussion. We

saw that annotations basically are metadata (data-oriented view), content about content (meta annotations) or additional content (content annotations) from an information-oriented view. Furthermore, they are communicative acts. From the examinations of annotations and annotation systems, a formal model of annotations, aimed at information retrieval, was derived. In this model, all objects which are deemed important for retrieval are contained. The act of annotation does not only produce the annotations themselves, but also additional objects like annotated fragments. Each annotation has at least one object it refers to, which is the annotation target. Furthermore, annotations as well as (multimedia) documents can be structured, so they need to be regarded as complex objects and their logical structure should be made explicit. The proposed model thus contains digital objects like documents, annotations, components and fragments, plus their typed interrelations (references, annotation targets). On the instance level, these objects form a structured annotation hypertext, which is an acyclic directed graph.

Part II gave an answer to the question how the annotation context can be modelled, queried and how annotation-based retrieval can be performed. It first discussed the general challenges of annotation-based retrieval, which are mainly document search (using annotations to find relevant documents) and discussion search (finding relevant annotations). The information contained in structured annotation hypertexts can potentially support sophisticated information needs. A logical framework called POLAR was defined which enables users to formulate complex queries to retrieve documents and annotations. With POLAR, structured annotation hypertexts can be modelled and queried. It allows for directly specifying the logical document structure, merged annotation targets, fragments, (content and meta level) annotations (which can be positive or negative) and references between objects. Annotation-based IR is realised by means of (knowledge and relevance) augmentation, which aggregates the several subcontexts related to annotation, and uncertain inference calculating the probability $P(d \rightarrow q)$ that a document (or annotation) d implies a query q. By applying four-valued logics, POLAR is able to cope with inconsistent knowledge which naturally arises in discussions. Propositions in POLAR can be terms, attributes and classifications; probabilities can be assigned to them as weights. This way, we cannot only realise probabilistic term weighting in POLAR, but by means of attributes and classifications, complex relationships between objects (needed, for instance, to represent hypermedia documents and annotations) and metadata can be provided. POLAR can be used to satisfy a variety of possible information needs, also incorporating non-topical information, as several application examples show. In the tradition of POOL, which is extended by POLAR, the semantics are based on possible worlds and Kripke structures. The original semantics of POOL needed to be expanded in order to deal with structured annotation hypertexts (and an underlying cyclic graph structure). POLAR programs are translated to four-valued probabilistic Datalog, which is then executed with an engine like HySpirit.

In Part III, the effectiveness of POLAR's core annotation-based retrieval approach, knowledge augmentation with a $tf \times idf$-based interpretation of $P(d \rightarrow q)$, was evaluated for discussion and document search. To evaluate discussion search, a test collection consisting of W3C email discussion lists was used. The new parts of emails were regarded as an annotation of its quoted parts, which belongs to the antecedent email. This way it was possible to extract merged annotation targets and fragments from emails. These extracted pieces and their interrelations were modelled in POLAR. For annotation-based document search, a new collection had to be created which contains both main documents and annotations. A snapshot of ZDNet News was harvested and represented in POLAR. Discussions in ZDNet are often very controversial, which makes the snapshot a good collection not only for document search, but also to explore methods which try to determine the polarity of annotations. The discussion search experiments showed

that fragments, merged targets and direct annotations are able to improve retrieval effectiveness significantly, depending on a suitable choice of access probabilities. A combination of these kinds of objects showed some further improvement. For document search, annotations helped improve the retrieval effectiveness if their content is propagated only moderately; it seems that radius-1 augmentation is sufficient. The document search experiments, but also the discussion search experiments, showed that the bias coming from the annotations must not be too high – we measure a decrease in retrieval effectiveness when the access probability is too high or when there are too many contextual objects contributing to the final retrieval status value. This is the case for the document search experiments, where articles tend to have many attached comments, but we can also observe this effect with discussion search. Besides the retrieval method, a machine-learning approach to detect the polarity of annotations was evaluated, which yielded an accuracy of 80%.

The annotation model in Chapter 3, which introduces documents, annotations, fragments, components and (indirectly through the hasAnnotationTarget relation) annotation targets is validated by the experimental results – (merged) targets, fragments and annotations can indeed influence the retrieval process in a positive way. The results also confirmed the integration of these elements into the POLAR framework and the application of the knowledge augmentation approach in conjunction with probabilistic inference. The evaluation also shows that annotations and their related objects indeed contain information which can be exploited for information retrieval. The hypothesis that annotations are a valuable source of evidence for retrieval is supported. This is one of the main findings of this thesis, besides the definition of an annotation model for retrieval and structured annotation hypertexts, and the specification of the POLAR framework as a flexible tool for annotation-based retrieval.

However, as annotations come in many forms and shapes and serve different purposes, the evaluation in this thesis can only be the beginning. In Chapter 4 we learnt about different application showcases of the POLAR framework. Many of them still need to be evaluated. For instance, evaluation initiatives like INEX showed that using the document structure is a good idea, but it is not clear whether annotations can aid structured document retrieval. Other experiments can address the effect of negative annotations, either with knowledge augmentation, which also considers the polarity of annotations, or by using the polarity to measure the trustworthiness of annotations, which is an important non-topical piece of evidence for determining the relevance of annotations. In general, diverse test collections are needed which contain different kinds of annotations, created with different tasks and purposes in mind – the test collections so far contained public annotations and discussions. Every user could potentially contribute to the discussion, which often spoils the quality of comments. This is especially the case with the ZDNet collection, where different opinions, for example about different operating systems like Linux or Windows, clash. Other annotation collections might be created by a different user group, for example if the discussion is only open for scientists. Furthermore, collections containing private, shared and public annotations, or coming from different domains with different tasks in mind could be an interesting data set for evaluation. Although such collections exists, they need to be set up to be a test collection, which requires huge efforts. Once this is done, POLAR could be a suitable tool for the evaluation of annotation-based retrieval.

While a machine-learning approach to determine the polarity of annotation has been evaluated, similar approaches might be used to distinguish meta from content annotations. Further evaluation could also apply flexible access probabilities, taking the number of subcontexts into

account. This way, we can better control the bias coming from annotations, merged targets and fragments for discussion and document search.

The syntax and semantics of POLAR are not carved in stone, but POLAR should be understood as a flexible system which is open for further enhancements. A possible extension regards the polarity of an annotation and its classification into a meta or content one. For future applications, we might want to model to what extent an annotation is positive or negative. In theory, annotations can even be both, at least they can contain positive and negative parts. Furthermore, we might want to incorporate the information to which degree an annotation is positive or negative. There is a similar case with content and meta annotations; an annotation might contain parts on the meta and parts on the content level, and future automatic classifiers might be able to detect to which degree an annotation is on the content or meta level. To express that in POLAR, it might be feasible to expand POLAR's syntax and semantics so that also the probability that a subcontext is a positive or negative meta or content annotation is reflected. Furthermore, within a structured annotation, it might be desirable to mark which subparts of an annotations are positive and negative on the content or meta level.

Appendix A

Model of the Annotation Universe

The next page shows the annotation universe expressed in Description Logics; see Chapter 3 for details.

$$
\begin{aligned}
\text{DigitalObject} &\equiv (= 1\ \text{hasURI}) \sqcap \forall \text{hasURI}.\texttt{URI} \sqcap \\
&\sqcap\ (\leq 1\ \text{hasCreationTime}) \sqcap \forall \text{hasCreationTime}.\texttt{Timestamp} \\
&\quad (\leq 1\ \text{hasBody}) \sqcap \forall \text{hasBody}.\texttt{String} \\
\text{AnnotatableObject} &\sqsubseteq \text{DigitalObject} \\
\text{Document} &\sqsubseteq \text{AnnotatableObject} \\
\text{Document} &\sqsubseteq (= 0\ \text{isPartOf}) \sqcap \\
&\quad (= 0\ \text{isFragmentOf}) \sqcap \\
&\quad (= 0\ \text{hasAnnotationTarget}) \sqcap \\
\text{Component} &\sqsubseteq \text{AnnotatableObject} \\
\text{Component} &\equiv (= 1\ \text{isPartOf}) \sqcap \\
&\quad \forall \text{isPartOf}.(\text{Document} \sqcup \text{Component} \sqcup \text{Annotation}) \\
\text{Component} &\sqsubseteq (= 0\ \text{isFragmentOf}) \sqcap \\
&\quad (= 0\ \text{hasAnnotationTarget}) \\
\text{Fragment} &\sqsubseteq \text{AnnotatableObject} \\
\text{Fragment} &\equiv (= 1\ \text{isFragmentOf}) \sqcap \\
&\quad \forall \text{isFragmentOf}.\text{AnnotatableObject} \\
\text{Fragment} &\sqsubseteq (= 0\ \text{isPartOf}) \sqcap \\
&\quad (= 0\ \text{hasAnnotationTarget}) \\
\text{Annotation} &\sqsubset \text{AnnotatableObject} \\
\text{Annotation} &\equiv (\geq 1\ \text{hasAnnotationTarget}) \sqcap \\
&\quad \forall \text{hasAnnotationTarget}.\text{AnnotatableObject} \sqcap \\
\text{Annotation} &\equiv (\geq 0\ \text{references}) \sqcap \\
&\quad \forall \text{references}.\text{DigitalObject} \\
\text{Annotation} &\sqsubseteq (= 1\ \text{hasAuthor}) \sqcap \forall \text{hasAuthor}.\text{User} \\
\text{Annotation} &\sqsubseteq (= 0\ \text{isPartOf}) \sqcap (= 0\ \text{isFragmentOf}) \\
\text{Annotation} &\sqsubseteq (\leq 1\ \text{polarity}) \sqcap \forall \text{polarity}.\texttt{String} \\
\text{Annotation} &\sqsubseteq (= 1\ \text{scope}) \sqcap \forall \text{scope}.\texttt{String}. \\
\text{Annotation} &\sqsubseteq (\geq 0\ \text{seenBy}) \sqcap \forall \text{seenBy}.\text{Group}. \\
\text{ContentLevelAnnotation} &\sqsubseteq \text{Annotation} \\
\text{MetaLevelAnnotation} &\sqsubseteq \text{Annotation} \\
\text{User} &\sqsubseteq (= 1\ \text{hasURI}) \sqcap \forall \text{hasURI}.\texttt{URI} \\
\text{Group} &\sqsubseteq (= 1\ \text{hasURI}) \sqcap \forall \text{hasURI}.\texttt{URI} \\
\text{User} &\sqsubseteq (\geq 0\ \text{isMemberOf}) \sqcap \forall \text{isMemberOf}.\text{Group} \\
\text{User} &\sqsubseteq \neg\text{Group} \sqcap \neg\text{DigitalObject} \\
\text{Group} &\sqsubseteq \neg\text{User} \sqcap \neg\text{DigitalObject}
\end{aligned}
$$

Appendix B

POLAR Implementation

B.1 FVPD Support Rules for Knowledge Augmentation

These FVPD rules need to be added to the FVPD translation of POLAR programs in order to enable knowledge augmenation:

```
######
# Augmentation rules for terms
######

term_k(T,D) :- term(T,D).
term_k(T,D) :- term_k_logical(T,D).
term_k(T,D) :- term_k_anno(T,D).
term_k(T,D) :- term_k_reference(T,D).

term_k_logical(T,D) :- acc_subpart(S,D) & term_k2(T,S).
term_k_anno(T,D) :- acc_mtarget(D,S) & term(T,S).
term_k_anno(T,D) :- acc_fragment(D,S) & term(T,S).
term_k_anno(T,D) :- acc_canno(D,S)  & term_k2(T,S).
term_k_anno(T,D) :- acc_negcanno(D,S) & !term_k2(T,S).
term_k_reference(T,D) :- acc_reference(D,S) & term_k2(T,S).



term_k2(T,D) :- term(T,D).
term_k2(T,D) :- term_k2_logical(T,D).
term_k2(T,D) :- term_k2_anno(T,D).
term_k2(T,D) :- term_k2_reference(T,D).

term_k2_logical(T,D) :- acc_subpart(D,S) & term_k2(T,S).
term_k2_anno(T,D) :- acc_fragment(D,S) & term(T,S).
term_k2_anno(T,D) :- acc_canno(D,S)  & term_k2(T,S).
term_k2_anno(T,D) :- acc_negcanno(D,S) & !term_k2(T,S).
term_k2_reference(T,D) :- acc_reference(D,S) & term_k2(T,S).

!term_k(T,D) :- !term(T,D).
!term_k(T,D) :- !term_k_logical(T,D).
```

```
31 !term_k(T,D) :- !term_k_anno(T,D).
32 !term_k(T,D) :- !term_k_reference(T,D).

34 !term_k_logical(T,D) :- acc_subpart(S,D) & !term_k2(T,S).
35 !term_k_anno(T,D) :- acc_mtarget(D,S) & !term(T,S).
36 !term_k_anno(T,D) :- acc_fragment(D,S) & !term(T,S).
37 !term_k_anno(T,D) :- acc_canno(D,S)  & !term_k2(T,S).
38 !term_k_anno(T,D) :- acc_negcanno(D,S) & term_k2(T,S).
39 !term_k_reference(T,D) :- acc_reference(D,S) & !term_k2(T,S).


42 !term_k2(T,D) :- !term(T,D).
43 !term_k2(T,D) :- !term_k2_logical(T,D).
44 !term_k2(T,D) :- !term_k2_anno(T,D).
45 !term_k2(T,D) :- !term_k2_reference(T,D).

47 !term_k2_logical(T,D) :- acc_subpart(D,S) & !term_k2(T,S).
48 !term_k2_anno(T,D) :- acc_fragment(D,S) & !term(T,S).
49 !term_k2_anno(T,D) :- acc_canno(D,S)  & !term_k2(T,S).
50 !term_k2_anno(T,D) :- acc_negcanno(D,S) & term_k2(T,S).
51 !term_k2_reference(T,D) :- acc_reference(D,S) & !term_k2(T,S).



55 ######
56 # Augmentation rules for categorisations
57 ######

59 instance_of_k(O,C,D) :- instance_of(O,C,D).
60 instance_of_k(O,C,D) :- instance_of_k_logical(O,C,D).
61 instance_of_k(O,C,D) :- instance_of_k_anno(O,C,D).
62 instance_of_k(O,C,D) :- instance_of_k_reference(O,C,D).

64 instance_of_k_logical(O,C,D) :-
65                 acc_subpart(S,D) & instance_of_k2(O,C,S).
66 instance_of_k_anno(O,C,D) :-
67                 acc_mtarget(D,S) & instance_of(O,C,S).
68 instance_of_k_anno(O,C,D) :-
69                 acc_fragment(D,S) & instance_of(O,C,S).
70 instance_of_k_anno(O,C,D) :-
71                 acc_canno(D,S)  & instance_of_k2(O,C,S).
72 instance_of_k_anno(O,C,D) :-
73                 acc_negcanno(D,S) & !instance_of_k2(O,C,S).
74 instance_of_k_reference(O,C,D) :-
75                 acc_reference(D,S) & instance_of_k2(O,C,S).


78 instance_of_k2(O,C,D) :- instance_of(O,C,D).
```

```
79 instance_of_k2(O,C,D) :- instance_of_k2_logical(O,C,D).
80 instance_of_k2(O,C,D) :- instance_of_k2_anno(O,C,D).
81 instance_of_k2(O,C,D) :- instance_of_k2_reference(O,C,D).

83 instance_of_k2_logical(O,C,D) :-
84                 acc_subpart(D,S) & instance_of_k2(O,C,S).
85 instance_of_k2_anno(O,C,D) :-
86                 acc_fragment(D,S) & instance_of(O,C,S).
87 instance_of_k2_anno(O,C,D) :-
88                 acc_canno(D,S)  & instance_of_k2(O,C,S).
89 instance_of_k2_anno(O,C,D) :-
90                 acc_negcanno(D,S) & !instance_of_k2(O,C,S).
91 instance_of_k2_reference(O,C,D) :-
92                 acc_reference(D,S) & instance_of_k2(O,C,S).


95 !instance_of_k(O,C,D) :- !instance_of(O,C,D).
96 !instance_of_k(O,C,D) :- !instance_of_k_logical(O,C,D).
97 !instance_of_k(O,C,D) :- !instance_of_k_anno(O,C,D).
98 !instance_of_k(O,C,D) :- !instance_of_k_reference(O,C,D).

100 !instance_of_k_logical(O,C,D) :-
101                 acc_subpart(S,D) & !instance_of_k2(O,C,S).
102 !instance_of_k_anno(O,C,D) :-
103                 acc_mtarget(D,S) & !instance_of(O,C,S).
104 !instance_of_k_anno(O,C,D) :-
105                 acc_fragment(D,S) & !instance_of(O,C,S).
106 !instance_of_k_anno(O,C,D) :-
107                 acc_canno(D,S)  & !instance_of_k2(O,C,S).
108 !instance_of_k_anno(O,C,D) :-
109                 acc_negcanno(D,S) & instance_of_k2(O,C,S).
110 !instance_of_k_reference(O,C,D) :-
111                 acc_reference(D,S) & !instance_of_k2(O,C,S).


114 !instance_of_k2(O,C,D) :- !instance_of(O,C,D).
115 !instance_of_k2(O,C,D) :- !instance_of_k2_logical(O,C,D).
116 !instance_of_k2(O,C,D) :- !instance_of_k2_anno(O,C,D).
117 !instance_of_k2(O,C,D) :- !instance_of_k2_reference(O,C,D).

119 !instance_of_k2_logical(O,C,D) :-
120                 acc_subpart(D,S) & !instance_of_k2(O,C,S).
121 !instance_of_k2_anno(O,C,D) :-
122                 acc_fragment(D,S) & !instance_of(O,C,S).
123 !instance_of_k2_anno(O,C,D) :-
124                 acc_canno(D,S)  & !instance_of_k2(O,C,S).
125 !instance_of_k2_anno(O,C,D) :-
126                 acc_negcanno(D,S) & instance_of_k2(O,C,S).
```

```
127 !instance_of_k2_reference(O,C,D) :-
128                 acc_reference(D,S) & !instance_of_k2(O,C,S).



132 ######
133 # Augmentation rules for attributes
134 ######

136 attribute_k(N,O,V,D) :- attribute(N,O,V,D).
137 attribute_k(N,O,V,D) :- attribute_k_logical(N,O,V,D).
138 attribute_k(N,O,V,D) :- attribute_k_anno(N,O,V,D).
139 attribute_k(N,O,V,D) :- attribute_k_reference(N,O,V,D).

141 attribute_k_logical(N,O,V,D) :-
142                 acc_subpart(S,D) & attribute_k2(N,O,V,S).
143 attribute_k_anno(N,O,V,D) :-
144                 acc_mtarget(D,S) & attribute(N,O,V,S).
145 attribute_k_anno(N,O,V,D) :-
146                 acc_fragment(D,S) & attribute(N,O,V,S).
147 attribute_k_anno(N,O,V,D) :-
148                 acc_canno(D,S)  & attribute_k2(N,O,V,S).
149 attribute_k_anno(N,O,V,D) :-
150                 acc_negcanno(D,S) & !attribute_k2(N,O,V,S).
151 attribute_k_reference(N,O,V,D) :-
152                 acc_reference(D,S) & attribute_k2(N,O,V,S).



155 attribute_k2(N,O,V,D) :- attribute(N,O,V,D).
156 attribute_k2(N,O,V,D) :- attribute_k2_logical(N,O,V,D).
157 attribute_k2(N,O,V,D) :- attribute_k2_anno(N,O,V,D).
158 attribute_k2(N,O,V,D) :- attribute_k2_reference(N,O,V,D).

160 attribute_k2_logical(N,O,V,D) :-
161                 acc_subpart(D,S) & attribute_k2(N,O,V,S).
162 attribute_k2_anno(N,O,V,D) :-
163                 acc_fragment(D,S) & attribute(N,O,V,S).
164 attribute_k2_anno(N,O,V,D) :-
165                 acc_canno(D,S)  & attribute_k2(N,O,V,S).
166 attribute_k2_anno(N,O,V,D) :-
167                 acc_negcanno(D,S) & !attribute_k2(N,O,V,S).
168 attribute_k2_reference(N,O,V,D) :-
169                 acc_reference(D,S) & attribute_k2(N,O,V,S).

171 !attribute_k(N,O,V,D) :- !attribute(N,O,V,D).
172 !attribute_k(N,O,V,D) :- !attribute_k_logical(N,O,V,D).
173 !attribute_k(N,O,V,D) :- !attribute_k_anno(N,O,V,D).
174 !attribute_k(N,O,V,D) :- !attribute_k_reference(N,O,V,D).
```

```
176 !attribute_k_logical(N,O,V,D) :-
177                 acc_subpart(S,D) & !attribute_k2(N,O,V,S).
178 !attribute_k_anno(N,O,V,D) :-
179                 acc_mtarget(D,S) & !attribute(N,O,V,S).
180 !attribute_k_anno(N,O,V,D) :-
181                 acc_fragment(D,S) & !attribute(N,O,V,S).
182 !attribute_k_anno(N,O,V,D) :-
183                 acc_canno(D,S)  & !attribute_k2(N,O,V,S).
184 !attribute_k_anno(N,O,V,D) :-
185                 acc_negcanno(D,S) & attribute_k2(N,O,V,S).
186 !attribute_k_reference(N,O,V,D) :-
187                 acc_reference(D,S) & !attribute_k2(N,O,V,S).


190 !attribute_k2(N,O,V,D) :- !attribute(N,O,V,D).
191 !attribute_k2(N,O,V,D) :- !attribute_k2_logical(N,O,V,D).
192 !attribute_k2(N,O,V,D) :- !attribute_k2_anno(N,O,V,D).
193 !attribute_k2(N,O,V,D) :- !attribute_k2_reference(N,O,V,D).

195 !attribute_k2_logical(N,O,V,D) :-
196                 acc_subpart(D,S) & !attribute_k2(N,O,V,S).
197 !attribute_k2_anno(N,O,V,D) :-
198                 acc_fragment(D,S) & !attribute(N,O,V,S).
199 !attribute_k2_anno(N,O,V,D) :-
200                 acc_canno(D,S)  & !attribute_k2(N,O,V,S).
201 !attribute_k2_anno(N,O,V,D) :-
202                 acc_negcanno(D,S) & attribute_k2(N,O,V,S).
203 !attribute_k2_reference(N,O,V,D) :-
204                 acc_reference(D,S) & !attribute_k2(N,O,V,S).
```

B.2 Calculation of η(pos_term_k(football,d1) & !neg_term_k(football,d1))

We show

$$\eta(\text{pos\_term\_k(football,d1) \& !neg\_term\_k(football,d1)}) = \\ \texttt{acc\_canno(d1,a1)} \wedge \texttt{term(t,football,a1)}$$

which is needed in Example 14 on page 145. We use the definition of event expressions and the function η for probabilistic Datalog as given in Fuhr (2000). We sometimes refer to the rules in the PD translation given in Example 14.

$$\eta(\text{pos\_term\_k(football,d1) \& !neg\_term\_k(football,D)}) = \\ \eta(\text{pos\_term\_k(football,d1)}) \wedge \eta(\text{!neg\_term\_k(football,D)})$$

Applying rules 11 to 14 and $\eta(!e) = \neg\eta(e)$ (Fuhr, 2000), this yields

η(pos_term_k(football,d1) :- acc_canno(d1,a1) &
pos_term_k(football,a1) & !neg_term_k(football,a1)) $\wedge$
$\neg\eta$(neg_term_k(football,d1) :- acc_canno(d1,a1) &
neg_term_k(football,a1) & !pos_term_k(football,a1)) =
η(acc_canno(d1,a1)) $\wedge$
η(pos_term_k(football,a1) $\wedge$ η(!neg_term_k(football,a1)) $\wedge$
$\neg$(η(acc_canno(d1,a1)) $\wedge$
η(neg_term_k(football,a1)) $\wedge$ η(!pos_term_k(football,a1)))

η(acc_canno(d1,a1)) = `acc_canno(d1,a1)`; together with applying rules 9 and 10, we get

`acc_canno(d1,a1)` $\wedge$
η(pos_term_k(football,a1) :-
pos_term(football,a1) & !neg_term(football,a1)) $\wedge$
$\neg\eta$(neg_term_k(football,a1) :-
neg_term(football,a1) & !pos_term(football,a1)) $\wedge$
$\neg$(`acc_canno(d1,a1)` $\wedge$
η(neg_term_k(football,a1) :-
neg_term(football,a1) & !pos_term(football,a1)) $\wedge$
$\neg\eta$(pos_term_k(football,a1) :-
pos_term(football,a1) & !neg_term(football,a1))) =
`acc_canno(d1,a1)` $\wedge$
η(pos_term(football,a1)) $\wedge$ η(!neg_term(football,a1)) $\wedge$
$\neg$(η(neg_term(football,a1)) $\wedge$ η(!pos_term(football,a1))) $\wedge$
$\neg$(`acc_canno(d1,a1)` $\wedge$
η(neg_term(football,a1)) $\wedge$ η(!pos_term(football,a1)) $\wedge$
$\neg$(η(pos_term(football,a1)) $\wedge$ η(!neg_term(football,a1))))

Rules 5 – 8 say, e.g., η(pos_term(football,a1)) = term(t,football,a1) ∨ term(i,football,a1) (analogously for η(neg_term(football,a1)), so we get

acc_canno(d1,a1) ∧
(term(t,football,a1) ∨ term(i,football,a1)) ∧
¬(term(f,football,a1) ∨ term(i,football,a1)) ∧
¬((term(f,football,a1) ∨ term(i,football,a1)) ∧
¬(term(t,football,a1) ∨ term(i,football,a1))) ∧
¬(acc_canno(d1,a1) ∧
(term(f,football,a1) ∨ term(i,football,a1)) ∧
¬(term(t,football,a1) ∨ term(i,football,a1)) ∧
¬((term(t,football,a1) ∨ term(i,football,a1)) ∧
¬(term(f,football,a1) ∨ term(i,football,a1))))

Because term(t,football,a1), term(f,football,a1) and term(i,football,a1) are disjoint events, it is, e.g., term(t,football,a1) ∧ ¬(term(f,football,a1) ∨ term(i,football,a1)) = term(t,football,a1) (see also (Fuhr, 2000, example 12)). Applying this and the distributive law[1], the above formula equals

acc_canno(d1,a1) ∧ term(t,football,a1) ∧ ¬term(f,football,a1) ∧
¬(acc_canno(d1,a1) ∧ term(f,football,a1) ∧ ¬term(t,football,a1)) =
acc_canno(d1,a1) ∧ term(t,football,a1) ∧
(¬acc_canno(d1,a1) ∨ ¬term(f,football,a1) ∨ term(t,football,a1))

Again we employ the disjointness of the term relation and the distributive law. We gain

¬acc_canno(d1,a1) ∧ acc_canno(d1,a1) ∧ term(t,football,a1) ∨
¬term(f,football,a1) ∧ acc_canno(d1,a1) ∧ term(t,football,a1) ∨
term(t,football,a1) ∧ acc_canno(d1,a1) ∧ term(t,football,a1) =
acc_canno(d1,a1) ∧ term(t,football,a1) ∨
acc_canno(d1,a1) ∧ term(t,football,a1) =
acc_canno(d1,a1) ∧ term(t,football,a1)

[1] $a \wedge (b \vee c) = a \wedge b \vee a \wedge c$

Appendix C

Further Evaluation Statistics

C.1 Ranking Statistics

Some statistics about the rankings produced in the discussion and document search experiments are reported here. For a certain feature, the number of documents with this feature, the number of relevant documents and the fraction of relevant documents to total number of documents are presented. For example, a feature could be "the number of fragments per article is greater than 0". Then, the entry "16.07/5.63 (0.35)" means that within the given number of articles (e.g., 30), there were on average 16.07 articles with at least 1 fragment. 5.63 of them were relevant, which makes a fraction of $\frac{5.63}{16.07} = 0.35$ of relevant to all documents (in other words: 35% of the documents with at least 1 fragment were relevant).

C.1.1 Discussion Search

The email features are the number of fragments per email (relevant for fragment access), which can be 0, >0, >5 and >10, and if an email has a merged target or not (relevant for merged target access). The following three pages present statistics for the top k ranked emails, with $k = 5, 10, 30$. Please refer to Section 8.3 for a further discussion of these statistics.

Fragments, merged targets and content annotations, top 5 emails considered:

Run	No merged target	One merged target
Baseline	2.75/1.12 (0.41)	2.25/1.30 (0.58)
knowlaug-mtarget-0.1	2.58/0.92 (0.35)	2.42/1.32 (0.54)
knowlaug-mtarget-0.2	2.42/1.00 (0.41)	2.58/1.52 (0.59)
knowlaug-mtarget-0.3	2.15/0.87 (0.40)	2.85/1.65 (0.58)
knowlaug-mtarget-0.4	1.88/0.80 (0.42)	3.12/1.75 (0.56)
knowlaug-mtarget-0.5	1.73/0.67 (0.38)	3.27/1.80 (0.55)
knowlaug-mtarget-0.6	1.52/0.62 (0.41)	3.48/1.88 (0.54)
knowlaug-mtarget-0.7	1.43/0.57 (0.40)	3.57/1.88 (0.53)
knowlaug-mtarget-0.8	1.25/0.50 (0.40)	3.75/1.87 (0.50)
knowlaug-mtarget-0.9	0.93/0.35 (0.37)	4.07/1.98 (0.49)
knowlaug-mtarget-1.0	0.77/0.30 (0.39)	4.23/2.05 (0.48)

	Number of fragments per email			
Run	0	>0	>5	>10
Baseline	3.70/1.62 (0.44)	1.30/0.80 (0.62)	0.00/0.00 (0.00)	0.00/0.00 (0.00)
knowlaug-fragments-0.1	3.42/1.50 (0.44)	1.58/0.95 (0.60)	0.02/0.02 (1.00)	0.00/0.00 (0.00)
knowlaug-fragments-0.2	3.02/1.37 (0.45)	1.98/1.17 (0.59)	0.03/0.03 (1.00)	0.00/0.00 (0.00)
knowlaug-fragments-0.3	2.72/1.23 (0.45)	2.28/1.32 (0.58)	0.12/0.07 (0.57)	0.02/0.00 (0.00)
knowlaug-fragments-0.4	2.42/1.10 (0.46)	2.58/1.40 (0.54)	0.17/0.07 (0.40)	0.02/0.00 (0.00)
knowlaug-fragments-0.5	2.20/1.05 (0.48)	2.80/1.42 (0.51)	0.20/0.08 (0.42)	0.02/0.00 (0.00)
knowlaug-fragments-0.6	1.92/0.92 (0.48)	3.08/1.53 (0.50)	0.22/0.08 (0.38)	0.03/0.00 (0.00)
knowlaug-fragments-0.7	1.57/0.80 (0.51)	3.43/1.62 (0.47)	0.27/0.07 (0.25)	0.03/0.00 (0.00)
knowlaug-fragments-0.8	1.30/0.67 (0.51)	3.70/1.73 (0.47)	0.25/0.07 (0.27)	0.03/0.00 (0.00)
knowlaug-fragments-0.9	1.12/0.58 (0.52)	3.88/1.82 (0.47)	0.25/0.07 (0.27)	0.03/0.00 (0.00)
knowlaug-fragments-1.0	0.93/0.57 (0.61)	4.07/1.82 (0.45)	0.25/0.07 (0.27)	0.03/0.00 (0.00)

Run	Number of direct comments per email			
	0	>0	>5	>10
Baseline	3.20/1.43 (0.45)	1.80/0.98 (0.55)	0.00/0.00 (0.00)	0.00/0.00 (0.00)
knowlaug-canno-0.1	2.78/1.28 (0.46)	2.22/1.28 (0.58)	0.03/0.03 (1.00)	0.00/0.00 (0.00)
knowlaug-canno-0.2	2.45/1.13 (0.46)	2.55/1.47 (0.58)	0.07/0.05 (0.75)	0.00/0.00 (0.00)
knowlaug-canno-0.3	2.15/1.03 (0.48)	2.85/1.62 (0.57)	0.17/0.08 (0.50)	0.02/0.00 (0.00)
knowlaug-canno-0.4	1.87/0.92 (0.49)	3.13/1.72 (0.55)	0.22/0.08 (0.38)	0.03/0.00 (0.00)
knowlaug-canno-0.5	1.52/0.67 (0.44)	3.48/1.85 (0.53)	0.32/0.10 (0.32)	0.05/0.00 (0.00)
knowlaug-canno-0.6	1.32/0.57 (0.43)	3.68/1.92 (0.52)	0.37/0.08 (0.23)	0.05/0.00 (0.00)
knowlaug-canno-0.7	1.12/0.52 (0.46)	3.88/2.05 (0.53)	0.42/0.10 (0.24)	0.05/0.00 (0.00)
knowlaug-canno-0.8	0.82/0.38 (0.47)	4.18/2.10 (0.50)	0.43/0.10 (0.23)	0.03/0.00 (0.00)
knowlaug-canno-0.9	0.63/0.27 (0.42)	4.37/2.10 (0.48)	0.42/0.10 (0.24)	0.03/0.00 (0.00)
knowlaug-canno-1.0	0.45/0.20 (0.44)	4.55/2.07 (0.45)	0.42/0.08 (0.20)	0.03/0.00 (0.00)

Fragments, merged targets and content annotations, top 10 emails considered:

Run	No merged target	One merged target
Baseline	5.30/1.78 (0.34)	4.70/2.62 (0.56)
knowlaug-mtarget-0.1	5.10/1.75 (0.34)	4.90/2.72 (0.55)
knowlaug-mtarget-0.2	4.82/1.67 (0.35)	5.18/2.82 (0.54)
knowlaug-mtarget-0.3	4.48/1.60 (0.36)	5.52/3.00 (0.54)
knowlaug-mtarget-0.4	4.12/1.47 (0.36)	5.88/3.05 (0.52)
knowlaug-mtarget-0.5	3.75/1.35 (0.36)	6.25/3.10 (0.50)
knowlaug-mtarget-0.6	3.37/1.25 (0.37)	6.63/3.32 (0.50)
knowlaug-mtarget-0.7	3.07/1.22 (0.40)	6.93/3.43 (0.50)
knowlaug-mtarget-0.8	2.60/1.05 (0.40)	7.40/3.53 (0.48)
knowlaug-mtarget-0.9	2.25/0.90 (0.40)	7.75/3.53 (0.46)
knowlaug-mtarget-1.0	1.92/0.77 (0.40)	8.08/3.60 (0.45)

	Number of fragments per email			
Run	0	>0	>5	>10
Baseline	7.12/2.62 (0.37)	2.88/1.78 (0.62)	0.00/0.00 (0.00)	0.00/0.00 (0.00)
knowlaug-fragment-0.1	6.83/2.52 (0.37)	3.17/1.90 (0.60)	0.03/0.02 (0.50)	0.00/0.00 (0.00)
knowlaug-fragment-0.2	6.30/2.40 (0.38)	3.70/2.13 (0.58)	0.12/0.07 (0.57)	0.00/0.00 (0.00)
knowlaug-fragment-0.3	5.73/2.30 (0.40)	4.27/2.27 (0.53)	0.20/0.08 (0.42)	0.02/0.00 (0.00)
knowlaug-fragment-0.4	5.23/2.15 (0.41)	4.77/2.42 (0.51)	0.22/0.10 (0.46)	0.02/0.00 (0.00)
knowlaug-fragment-0.5	4.80/2.05 (0.43)	5.20/2.48 (0.48)	0.28/0.10 (0.35)	0.05/0.00 (0.00)
knowlaug-fragment-0.6	4.15/1.72 (0.41)	5.85/2.62 (0.45)	0.35/0.10 (0.29)	0.05/0.00 (0.00)
knowlaug-fragment-0.7	3.53/1.58 (0.45)	6.47/2.83 (0.44)	0.33/0.10 (0.30)	0.05/0.00 (0.00)
knowlaug-fragment-0.8	3.10/1.47 (0.47)	6.90/2.97 (0.43)	0.35/0.10 (0.29)	0.05/0.00 (0.00)
knowlaug-fragment-0.9	2.72/1.33 (0.49)	7.28/2.93 (0.40)	0.37/0.10 (0.27)	0.05/0.00 (0.00)
knowlaug-fragment-1.0	2.38/1.23 (0.52)	7.62/2.97 (0.39)	0.37/0.10 (0.27)	0.05/0.00 (0.00)

Run	Number of direct comments per email			
	0	>0	>5	>10
Baseline	6.22/2.30 (0.37)	3.78/2.10 (0.56)	0.00/0.00 (0.00)	0.00/0.00 (0.00)
knowlaug-canno-0.1	5.68/2.17 (0.38)	4.32/2.45 (0.57)	0.05/0.03 (0.67)	0.00/0.00 (0.00)
knowlaug-canno-0.2	5.08/2.02 (0.40)	4.92/2.85 (0.58)	0.15/0.10 (0.67)	0.00/0.00 (0.00)
knowlaug-canno-0.3	4.58/1.90 (0.41)	5.42/3.05 (0.56)	0.28/0.10 (0.35)	0.03/0.00 (0.00)
knowlaug-canno-0.4	4.05/1.72 (0.42)	5.95/3.23 (0.54)	0.43/0.15 (0.35)	0.07/0.00 (0.00)
knowlaug-canno-0.5	3.52/1.55 (0.44)	6.48/3.30 (0.51)	0.50/0.15 (0.30)	0.07/0.00 (0.00)
knowlaug-canno-0.6	2.95/1.40 (0.47)	7.05/3.47 (0.49)	0.57/0.13 (0.24)	0.07/0.00 (0.00)
knowlaug-canno-0.7	2.48/1.17 (0.47)	7.52/3.48 (0.46)	0.60/0.13 (0.22)	0.05/0.00 (0.00)
knowlaug-canno-0.8	2.00/0.93 (0.47)	8.00/3.53 (0.44)	0.62/0.13 (0.22)	0.05/0.00 (0.00)
knowlaug-canno-0.9	1.67/0.87 (0.52)	8.33/3.47 (0.42)	0.60/0.12 (0.19)	0.05/0.00 (0.00)
knowlaug-canno-1.0	1.33/0.80 (0.60)	8.67/3.43 (0.40)	0.63/0.12 (0.18)	0.03/0.00 (0.00)

Fragments, merged targets and content annotations, top 30 emails considered:

Run	No merged target	One merged target
Baseline	16.52/4.32 (0.26)	13.48/5.52 (0.41)
knowlaug-mtarget-0.1	16.05/4.23 (0.26)	13.95/5.85 (0.42)
knowlaug-mtarget-0.2	15.45/4.15 (0.27)	14.55/6.10 (0.42)
knowlaug-mtarget-0.3	14.67/3.95 (0.27)	15.33/6.40 (0.42)
knowlaug-mtarget-0.4	13.88/3.85 (0.28)	16.12/6.55 (0.41)
knowlaug-mtarget-0.5	12.87/3.53 (0.27)	17.13/6.80 (0.40)
knowlaug-mtarget-0.6	11.67/3.28 (0.28)	18.33/7.23 (0.39)
knowlaug-mtarget-0.7	10.48/3.03 (0.29)	19.52/7.43 (0.38)
knowlaug-mtarget-0.8	9.30/2.72 (0.29)	20.70/7.65 (0.37)
knowlaug-mtarget-0.9	8.13/2.40 (0.30)	21.87/7.77 (0.36)
knowlaug-mtarget-1.0	6.85/2.22 (0.32)	23.15/7.63 (0.33)

	Number of fragments per email			
Run	0	>0	>5	>10
Baseline	21.63/6.07 (0.28)	8.37/3.77 (0.45)	0.08/0.07 (0.80)	0.00/0.00 (0.00)
knowlaug-fragments-0.1	20.55/5.85 (0.28)	9.45/4.08 (0.43)	0.12/0.08 (0.71)	0.00/0.00 (0.00)
knowlaug-fragments-0.2	19.48/5.63 (0.29)	10.52/4.48 (0.43)	0.27/0.10 (0.38)	0.02/0.00 (0.00)
knowlaug-fragments-0.3	18.10/5.15 (0.28)	11.90/4.75 (0.40)	0.33/0.12 (0.35)	0.02/0.00 (0.00)
knowlaug-fragments-0.4	16.38/4.80 (0.29)	13.62/4.98 (0.37)	0.55/0.13 (0.24)	0.07/0.00 (0.00)
knowlaug-fragments-0.5	14.88/4.43 (0.30)	15.12/5.22 (0.35)	0.65/0.15 (0.23)	0.07/0.00 (0.00)
knowlaug-fragments-0.6	13.62/4.23 (0.31)	16.38/5.33 (0.33)	0.78/0.17 (0.21)	0.08/0.00 (0.00)
knowlaug-fragments-0.7	12.23/3.87 (0.32)	17.77/5.33 (0.30)	0.88/0.17 (0.19)	0.07/0.00 (0.00)
knowlaug-fragments-0.8	10.97/3.60 (0.33)	19.03/5.37 (0.28)	0.92/0.20 (0.22)	0.07/0.00 (0.00)
knowlaug-fragments-0.9	9.80/3.37 (0.34)	20.20/5.47 (0.27)	0.98/0.20 (0.20)	0.07/0.00 (0.00)
knowlaug-fragments-1.0	8.82/3.17 (0.36)	21.18/5.58 (0.26)	1.00/0.20 (0.20)	0.08/0.00 (0.00)

Run	Number of direct comments per email			
	0	>0	>5	>10
Baseline	18.82/5.18 (0.28)	11.18/4.65 (0.42)	0.08/0.07 (0.80)	0.00/0.00 (0.00)
knowlaug-canno-0.1	17.62/4.87 (0.28)	12.38/5.27 (0.43)	0.15/0.08 (0.56)	0.00/0.00 (0.00)
knowlaug-canno-0.2	16.05/4.57 (0.28)	13.95/5.77 (0.41)	0.42/0.12 (0.28)	0.07/0.00 (0.00)
knowlaug-canno-0.3	14.70/4.12 (0.28)	15.30/6.23 (0.41)	0.57/0.18 (0.32)	0.07/0.00 (0.00)
knowlaug-canno-0.4	12.92/3.65 (0.28)	17.08/6.70 (0.39)	0.77/0.20 (0.26)	0.07/0.00 (0.00)
knowlaug-canno-0.5	11.53/3.43 (0.30)	18.47/6.87 (0.37)	0.93/0.23 (0.25)	0.07/0.00 (0.00)
knowlaug-canno-0.6	10.10/3.17 (0.31)	19.90/7.17 (0.36)	1.03/0.23 (0.23)	0.10/0.00 (0.00)
knowlaug-canno-0.7	8.85/2.95 (0.33)	21.15/7.33 (0.35)	1.13/0.22 (0.19)	0.10/0.00 (0.00)
knowlaug-canno-0.8	7.65/2.67 (0.35)	22.35/7.20 (0.32)	1.25/0.22 (0.17)	0.12/0.00 (0.00)
knowlaug-canno-0.9	6.42/2.40 (0.37)	23.58/7.15 (0.30)	1.43/0.18 (0.13)	0.12/0.00 (0.00)
knowlaug-canno-1.0	5.33/2.07 (0.39)	24.67/7.15 (0.29)	1.42/0.15 (0.11)	0.10/0.00 (0.00)

C.1.2 Document Search

For document search, the ZDNet article features are the number of direct comments per article, which can be 0, >0, >5, >10,>50,>100 and >200. The following three tables present statistics for the top k ranked articles, with $k = 5, 10, 30$. Refer to Section 8.4 for a further discussion of these statistics.

Top 5 articles considered:

	Number of direct comments per article						
Run	0	>0	>5	>10	>50	>100	>200
Baseline	0.78/0.48 (0.61)	4.22/3.00 (0.71)	2.78/1.96 (0.70)	1.70/1.17 (0.69)	0.22/0.17 (0.80)	0.04/0.04 (1.00)	0.00/0.00 (0.00)
knowlaug-0.1	0.61/0.35 (0.57)	4.39/3.17 (0.72)	3.22/2.30 (0.72)	2.30/1.61 (0.70)	0.91/0.65 (0.71)	0.30/0.22 (0.71)	0.13/0.09 (0.67)
knowlaug-0.2	0.39/0.26 (0.67)	4.61/3.09 (0.67)	3.65/2.43 (0.67)	2.83/1.83 (0.65)	1.22/0.74 (0.61)	0.43/0.22 (0.50)	0.22/0.09 (0.40)
knowlaug-0.3	0.35/0.22 (0.62)	4.65/3.13 (0.67)	3.78/2.57 (0.68)	2.91/1.91 (0.66)	1.26/0.74 (0.59)	0.48/0.22 (0.45)	0.26/0.09 (0.33)
knowlaug-0.4	0.26/0.22 (0.83)	4.74/3.17 (0.67)	3.96/2.61 (0.66)	3.13/1.96 (0.62)	1.30/0.74 (0.57)	0.48/0.17 (0.36)	0.26/0.09 (0.33)
knowlaug-0.5	0.17/0.17 (1.00)	4.83/3.04 (0.63)	4.13/2.52 (0.61)	3.26/1.91 (0.59)	1.48/0.70 (0.47)	0.61/0.17 (0.29)	0.30/0.04 (0.14)
knowlaug-0.6	0.17/0.17 (1.00)	4.83/2.91 (0.60)	4.26/2.48 (0.58)	3.57/2.00 (0.56)	1.70/0.74 (0.44)	0.78/0.22 (0.28)	0.26/0.04 (0.17)
knowlaug-0.7	0.09/0.09 (1.00)	4.91/2.91 (0.59)	4.35/2.48 (0.57)	3.70/2.04 (0.55)	1.83/0.78 (0.43)	0.91/0.22 (0.24)	0.39/0.09 (0.22)
knowlaug-0.8	0.04/0.04 (1.00)	4.96/2.74 (0.55)	4.52/2.43 (0.54)	3.87/2.00 (0.52)	1.83/0.70 (0.38)	0.91/0.17 (0.19)	0.35/0.04 (0.12)
knowlaug-0.9	0.04/0.04 (1.00)	4.96/2.70 (0.54)	4.65/2.43 (0.52)	4.00/2.04 (0.51)	1.78/0.70 (0.39)	0.83/0.17 (0.21)	0.35/0.04 (0.12)
knowlaug-1.0	0.04/0.04 (1.00)	4.96/2.43 (0.49)	4.57/2.17 (0.48)	4.04/1.87 (0.46)	1.57/0.48 (0.31)	0.65/0.13 (0.20)	0.30/0.04 (0.14)
knowlaug-r1-0.1	0.61/0.35 (0.57)	4.39/3.17 (0.72)	3.22/2.30 (0.72)	2.26/1.57 (0.69)	0.78/0.57 (0.72)	0.26/0.17 (0.67)	0.09/0.04 (0.50)
knowlaug-r1-0.2	0.48/0.26 (0.55)	4.52/3.13 (0.69)	3.57/2.48 (0.70)	2.74/1.87 (0.68)	1.17/0.74 (0.63)	0.39/0.22 (0.56)	0.17/0.09 (0.50)
knowlaug-r1-0.3	0.39/0.22 (0.56)	4.61/3.04 (0.66)	3.70/2.43 (0.66)	2.83/1.78 (0.63)	1.22/0.70 (0.57)	0.43/0.22 (0.50)	0.22/0.09 (0.40)
knowlaug-r1-0.4	0.26/0.22 (0.83)	4.74/2.91 (0.61)	3.87/2.35 (0.61)	3.04/1.74 (0.57)	1.35/0.70 (0.52)	0.52/0.22 (0.42)	0.26/0.09 (0.33)
knowlaug-r1-0.5	0.22/0.17 (0.80)	4.78/2.87 (0.60)	4.00/2.35 (0.59)	3.13/1.78 (0.57)	1.43/0.74 (0.52)	0.57/0.22 (0.38)	0.30/0.09 (0.29)
knowlaug-r1-0.6	0.17/0.17 (1.00)	4.83/2.96 (0.61)	4.04/2.43 (0.60)	3.22/1.87 (0.58)	1.52/0.74 (0.49)	0.61/0.17 (0.29)	0.26/0.04 (0.17)
knowlaug-r1-0.7	0.17/0.17 (1.00)	4.83/2.87 (0.59)	4.17/2.43 (0.58)	3.39/1.91 (0.56)	1.70/0.78 (0.46)	0.70/0.22 (0.31)	0.22/0.04 (0.20)
knowlaug-r1-0.8	0.09/0.09 (1.00)	4.91/2.83 (0.58)	4.30/2.43 (0.57)	3.48/1.87 (0.54)	1.78/0.74 (0.41)	0.78/0.17 (0.22)	0.22/0.04 (0.20)
knowlaug-r1-0.9	0.09/0.09 (1.00)	4.91/2.65 (0.54)	4.39/2.35 (0.53)	3.57/1.83 (0.51)	1.91/0.74 (0.39)	0.87/0.17 (0.20)	0.26/0.04 (0.17)
knowlaug-r1-1.0	0.09/0.09 (1.00)	4.91/2.65 (0.54)	4.35/2.30 (0.53)	3.57/1.83 (0.51)	1.87/0.65 (0.35)	0.87/0.17 (0.20)	0.26/0.04 (0.17)

Top 10 articles considered:

	Number of direct comments per article						
Run	0	>0	>5	>10	>50	>100	>200
Baseline	1.70/0.91 (0.54)	8.30/5.48 (0.66)	4.91/3.17 (0.65)	3.04/1.91 (0.63)	0.35/0.26 (0.75)	0.09/0.04 (0.50)	0.00/0.00 (0.00)
knowlaug-0.1	1.26/0.74 (0.59)	8.74/5.70 (0.65)	6.04/3.83 (0.63)	4.30/2.70 (0.63)	1.13/0.83 (0.73)	0.30/0.22 (0.71)	0.13/0.09 (0.67)
knowlaug-0.2	1.00/0.61 (0.61)	9.00/5.78 (0.64)	6.83/4.22 (0.62)	5.17/3.17 (0.61)	1.78/1.04 (0.59)	0.70/0.35 (0.50)	0.30/0.09 (0.29)
knowlaug-0.3	0.74/0.48 (0.65)	9.26/5.96 (0.64)	7.22/4.39 (0.61)	5.57/3.30 (0.59)	2.00/1.04 (0.52)	0.78/0.35 (0.44)	0.35/0.09 (0.25)
knowlaug-0.4	0.65/0.39 (0.60)	9.35/5.70 (0.61)	7.48/4.30 (0.58)	5.78/3.17 (0.55)	2.52/1.09 (0.43)	1.22/0.35 (0.29)	0.43/0.09 (0.20)
knowlaug-0.5	0.48/0.35 (0.73)	9.52/5.65 (0.59)	7.61/4.30 (0.57)	5.87/3.13 (0.53)	2.52/1.04 (0.41)	1.26/0.30 (0.24)	0.43/0.09 (0.20)
knowlaug-0.6	0.39/0.30 (0.78)	9.61/5.57 (0.58)	7.61/4.22 (0.55)	5.91/3.04 (0.51)	2.70/1.04 (0.39)	1.35/0.30 (0.23)	0.48/0.09 (0.18)
knowlaug-0.7	0.30/0.26 (0.86)	9.70/5.13 (0.53)	8.04/4.00 (0.50)	6.43/3.00 (0.47)	3.13/1.04 (0.33)	1.52/0.30 (0.20)	0.52/0.09 (0.17)
knowlaug-0.8	0.17/0.17 (1.00)	9.83/4.83 (0.49)	8.30/3.83 (0.46)	6.78/2.91 (0.43)	3.26/1.00 (0.31)	1.52/0.26 (0.17)	0.48/0.09 (0.18)
knowlaug-0.9	0.17/0.17 (1.00)	9.83/4.26 (0.43)	8.61/3.57 (0.41)	7.26/2.83 (0.39)	3.43/0.87 (0.25)	1.57/0.22 (0.14)	0.48/0.09 (0.18)
knowlaug-1.0	0.13/0.13 (1.00)	9.87/4.04 (0.41)	8.74/3.43 (0.39)	7.43/2.70 (0.36)	3.09/0.70 (0.23)	1.43/0.22 (0.15)	0.43/0.09 (0.20)
knowlaug-r1-0.1	1.26/0.74 (0.59)	8.74/5.74 (0.66)	6.04/3.87 (0.64)	4.26/2.70 (0.63)	1.13/0.83 (0.73)	0.30/0.22 (0.71)	0.13/0.09 (0.67)
knowlaug-r1-0.2	1.04/0.61 (0.58)	8.96/5.83 (0.65)	6.74/4.22 (0.63)	5.04/3.09 (0.61)	1.70/1.00 (0.59)	0.70/0.35 (0.50)	0.30/0.09 (0.29)
knowlaug-r1-0.3	0.83/0.52 (0.63)	9.17/5.83 (0.64)	7.04/4.26 (0.60)	5.43/3.22 (0.59)	2.00/1.09 (0.54)	0.78/0.35 (0.44)	0.35/0.09 (0.25)
knowlaug-r1-0.4	0.70/0.43 (0.62)	9.30/5.70 (0.61)	7.39/4.26 (0.58)	5.70/3.13 (0.55)	2.30/1.04 (0.45)	1.00/0.35 (0.35)	0.43/0.09 (0.20)
knowlaug-r1-0.5	0.65/0.39 (0.60)	9.35/5.52 (0.59)	7.43/4.17 (0.56)	5.74/3.04 (0.53)	2.52/1.04 (0.41)	1.22/0.35 (0.29)	0.48/0.09 (0.18)
knowlaug-r1-0.6	0.52/0.35 (0.67)	9.48/5.52 (0.58)	7.52/4.13 (0.55)	5.78/3.00 (0.52)	2.65/1.09 (0.41)	1.30/0.35 (0.27)	0.48/0.09 (0.18)
knowlaug-r1-0.7	0.43/0.35 (0.80)	9.57/5.57 (0.58)	7.57/4.17 (0.55)	5.83/3.04 (0.52)	2.65/1.04 (0.39)	1.22/0.30 (0.25)	0.39/0.09 (0.22)
knowlaug-r1-0.8	0.35/0.30 (0.88)	9.65/5.48 (0.57)	7.78/4.17 (0.54)	5.96/3.00 (0.50)	2.78/1.04 (0.38)	1.39/0.30 (0.22)	0.39/0.09 (0.22)
knowlaug-r1-0.9	0.35/0.30 (0.88)	9.65/5.35 (0.55)	7.78/4.13 (0.53)	6.09/3.00 (0.49)	2.87/1.04 (0.36)	1.39/0.30 (0.22)	0.39/0.09 (0.22)
knowlaug-r1-1.0	0.17/0.17 (1.00)	9.83/5.26 (0.54)	7.83/4.00 (0.51)	6.17/2.91 (0.47)	2.96/1.04 (0.35)	1.39/0.30 (0.22)	0.39/0.09 (0.22)

Top 30 articles considered:

	Number of direct comments per article						
Run	0	>0	>5	>10	>50	>100	>200
Baseline	5.09/2.13 (0.42)	24.91/12.78 (0.51)	14.35/7.57 (0.53)	9.22/5.00 (0.54)	1.48/1.09 (0.74)	0.35/0.17 (0.50)	0.00/0.00 (0.00)
knowlaug-0.1	4.48/2.00 (0.45)	25.52/13.04 (0.51)	15.70/8.09 (0.52)	11.09/5.70 (0.51)	2.48/1.39 (0.56)	0.96/0.35 (0.36)	0.26/0.09 (0.33)
knowlaug-0.2	4.00/1.78 (0.45)	26.00/12.91 (0.50)	16.87/8.09 (0.48)	12.48/5.74 (0.46)	3.70/1.48 (0.40)	1.57/0.43 (0.28)	0.43/0.09 (0.20)
knowlaug-0.3	3.52/1.70 (0.48)	26.48/12.48 (0.47)	18.30/8.09 (0.44)	13.96/5.74 (0.41)	4.78/1.48 (0.31)	2.04/0.43 (0.21)	0.48/0.09 (0.18)
knowlaug-0.4	3.09/1.57 (0.51)	26.91/12.13 (0.45)	19.52/8.17 (0.42)	15.35/5.91 (0.39)	5.43/1.48 (0.27)	2.43/0.43 (0.18)	0.52/0.09 (0.17)
knowlaug-0.5	2.61/1.39 (0.53)	27.39/11.70 (0.43)	20.91/8.09 (0.39)	16.70/5.83 (0.35)	6.43/1.48 (0.23)	3.04/0.43 (0.14)	0.65/0.09 (0.13)
knowlaug-0.6	2.17/1.09 (0.50)	27.83/11.30 (0.41)	21.87/7.87 (0.36)	17.70/5.70 (0.32)	7.00/1.48 (0.21)	3.30/0.43 (0.13)	0.74/0.09 (0.12)
knowlaug-0.7	1.74/0.87 (0.50)	28.26/11.04 (0.39)	22.87/7.83 (0.34)	18.78/5.74 (0.31)	7.48/1.48 (0.20)	3.39/0.43 (0.13)	0.78/0.09 (0.11)
knowlaug-0.8	1.39/0.61 (0.44)	28.61/10.30 (0.36)	23.65/7.17 (0.30)	19.87/5.39 (0.27)	8.17/1.39 (0.17)	3.70/0.43 (0.12)	0.87/0.09 (0.10)
knowlaug-0.9	0.96/0.48 (0.50)	29.04/9.43 (0.32)	24.35/6.78 (0.28)	20.65/5.13 (0.25)	8.43/1.39 (0.16)	3.74/0.43 (0.12)	0.83/0.09 (0.11)
knowlaug-1.0	0.52/0.35 (0.67)	29.48/8.74 (0.30)	25.17/6.61 (0.26)	21.13/5.04 (0.24)	8.13/1.30 (0.16)	3.57/0.39 (0.11)	0.83/0.09 (0.11)
knowlaug-r1-0.1	4.57/2.00 (0.44)	25.43/13.04 (0.51)	15.61/8.09 (0.52)	10.96/5.70 (0.52)	2.43/1.39 (0.57)	0.91/0.35 (0.38)	0.26/0.09 (0.33)
knowlaug-r1-0.2	4.04/1.78 (0.44)	25.96/13.00 (0.50)	16.78/8.22 (0.49)	12.30/5.87 (0.48)	3.48/1.48 (0.42)	1.48/0.43 (0.29)	0.39/0.09 (0.22)
knowlaug-r1-0.3	3.65/1.74 (0.48)	26.35/12.74 (0.48)	17.78/8.13 (0.46)	13.26/5.78 (0.44)	4.35/1.48 (0.34)	1.74/0.43 (0.25)	0.52/0.09 (0.17)
knowlaug-r1-0.4	3.39/1.61 (0.47)	26.61/12.30 (0.46)	18.74/8.13 (0.43)	14.39/5.87 (0.41)	4.96/1.48 (0.30)	2.22/0.43 (0.20)	0.57/0.09 (0.15)
knowlaug-r1-0.5	3.04/1.57 (0.51)	26.96/12.30 (0.46)	19.52/8.22 (0.42)	15.13/5.96 (0.39)	5.35/1.48 (0.28)	2.48/0.43 (0.18)	0.61/0.09 (0.14)
knowlaug-r1-0.6	2.78/1.52 (0.55)	27.22/12.09 (0.44)	20.30/8.26 (0.41)	15.91/5.96 (0.37)	5.83/1.48 (0.25)	2.70/0.43 (0.16)	0.61/0.09 (0.14)
knowlaug-r1-0.7	2.48/1.30 (0.53)	27.52/11.87 (0.43)	21.13/8.17 (0.39)	16.65/5.87 (0.35)	6.39/1.48 (0.23)	3.00/0.43 (0.14)	0.74/0.09 (0.12)
knowlaug-r1-0.8	2.13/1.04 (0.49)	27.87/11.39 (0.41)	21.74/7.87 (0.36)	17.43/5.65 (0.32)	6.87/1.48 (0.22)	3.35/0.43 (0.13)	0.96/0.09 (0.09)
knowlaug-r1-0.9	1.96/0.96 (0.49)	28.04/11.09 (0.40)	22.30/7.78 (0.35)	18.09/5.65 (0.31)	7.39/1.48 (0.20)	3.61/0.43 (0.12)	0.96/0.09 (0.09)
knowlaug-r1-1.0	1.78/0.83 (0.46)	28.22/10.91 (0.39)	22.78/7.70 (0.34)	18.52/5.57 (0.30)	7.52/1.48 (0.20)	3.61/0.43 (0.12)	0.96/0.09 (0.09)

C.2 Recall-Precision-Graphs

C.2.1 Discussion Search

The recall-precision graphs of the discussion search runs are shown here.

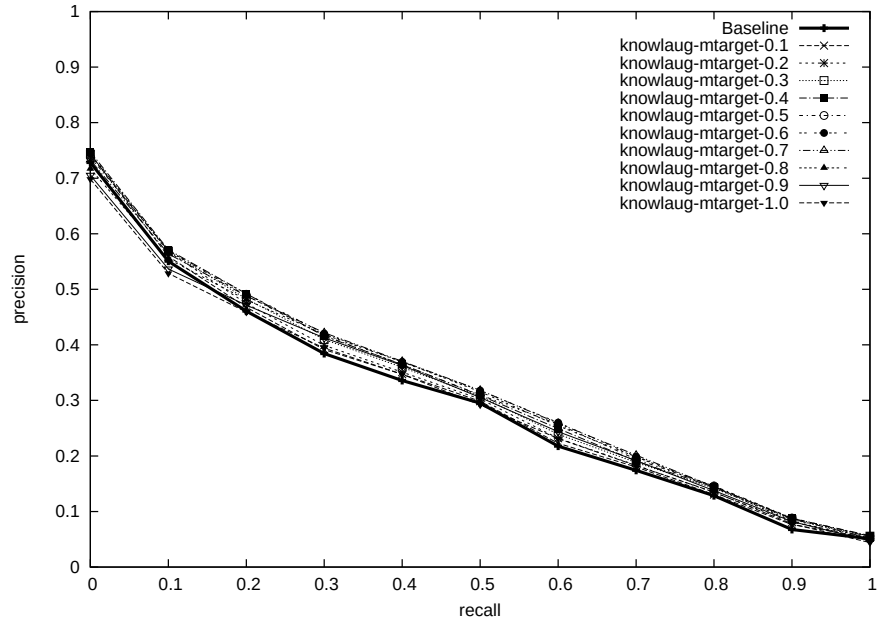


Figure C.1: Recall-precision graph of all knowledge augmentation merged target runs

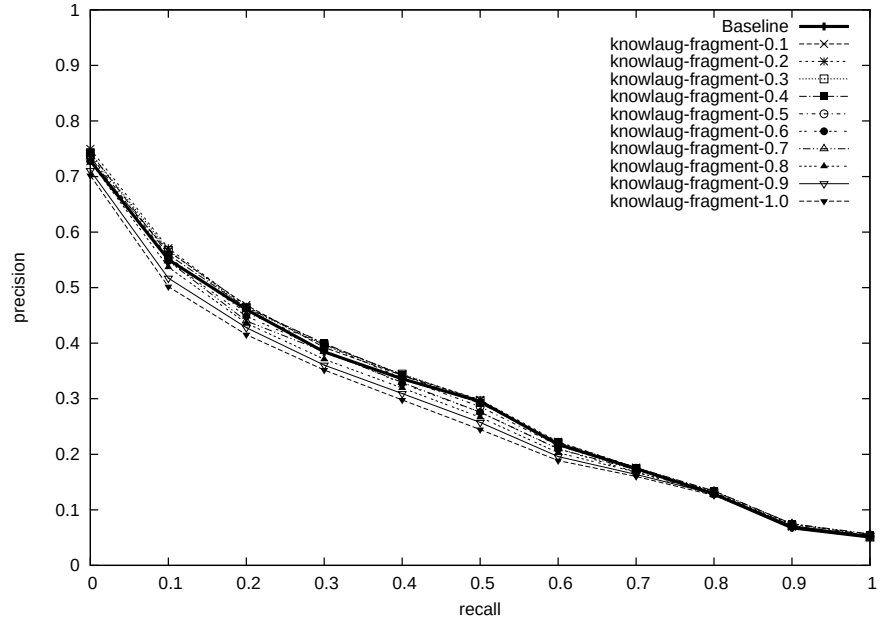


Figure C.2: Recall-precision graph of all knowledge augmentation fragment runs

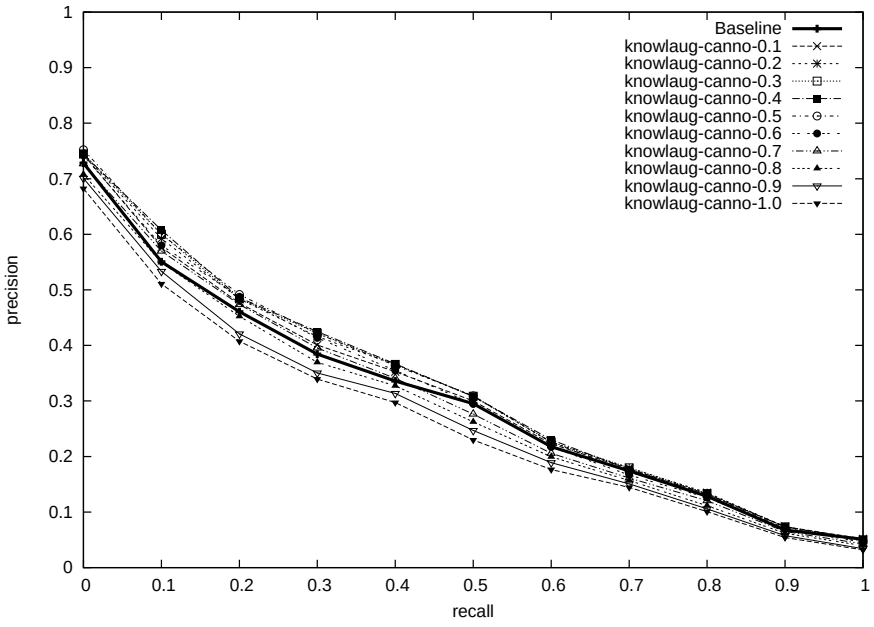


Figure C.3: Recall-precision graph of all knowledge augmentation content annotation runs

List of Definitions

List of Examples

List of Propositions

List of Figures

List of Tables

Index

Bibliography

Abolhassani, M. and Fuhr, N. (2004). Applying the divergence from randomness approach for content-only search in XML documents. In *26th European Conference on Information Retrieval Research (ECIR 2004)*, Heidelberg et al. Springer.

Agosti, M. (1996). An overview of hypertext. In Agosti and Smeaton (1996), chapter 2.

Agosti, M., Albrechtsen, H., Ferro, N., Frommholz, I., Hansen, P., Orio, N., Panizzi, E., Pejtersen, A. M., and Thiel, U. (2005a). DiLAS: a digital library annotation service. In *Proceedings of Annotation for Collaboration – A Workshop on Annotation Models, Tools and Practices*.

Agosti, M., Bonfiglio-Dosio, G., and Ferro, N. (2007a). A historical and contemporary study on annotations to derive key features for systems design. *International Journal on Digital Libraries (IJDL)*, 8(1):1–19.

Agosti, M., Coppotelli, T., Ferro, N., and Pretto, L. (2007b). Annotations and digital libraries: Designing adequate test-beds. In Goh, D. H.-L., Cao, T. H., Sølvberg, I., and Rasmussen, E. M., editors, *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers, 10th International Conference on Asian Digital Libraries, ICADL 2007, Hanoi, Vietnam, December 10-13, 2007, Proceedings*, volume 4822 of *Lecture Notes in Computer Science*, pages 150–159. Springer.

Agosti, M. and Ferro, N. (2003). Annotations: Enriching a digital library. In Constantopoulos and Sølvberg (2003), pages 88–100.

Agosti, M. and Ferro, N. (2005). Annotations as context for searching documents. In Crestani, F. and Ruthven, I., editors, *Information Context: Nature, Impact, and Role: 5th International Conference on Conceptions of Library and Information Sciences, CoLIS 2005*, volume 3507 of *Lecture Notes in Computer Science*, pages 155–170, Heidelberg et al. Springer.

Agosti, M. and Ferro, N. (2006). Search strategies for finding annotations and annotated documents: the FAST service. In Larsen, H. L., Pasi, G., Arroyo, D. O., Andreasen, T., and Christiansen, H., editors, *Proc. of the 7th International Conference on Flexible Querying Answering Systems (FQAS 2006)*, volume 4027 of *Lecture Notes in Computer Science*, pages 270–281, Heidelberg et al. Springer.

Agosti, M. and Ferro, N. (2007). A formal model of annotations of digital content. *ACM Transactions on Information Systems*, 26(13.):3.1 – 3.57.

Agosti, M., Ferro, N., Frommholz, I., Panizzi, E., Putz, W., and Thiel, U. (2006). Integration of the DiLAS annotation service into digital library infrastructures. In Blandford, A. and

Gow, J., editors, *Proc. Digital Libraries in the Context of Users' Broader Activities Workshop (CUBA) at JCDL 2006*.

Agosti, M., Ferro, N., Frommholz, I., and Thiel, U. (2004). Annotations in digital libraries and collaboratories – facets, models and usage. In Heery and Lyon (2004), pages 244–255.

Agosti, M., Ferro, N., and Orio, N. (2005b). Annotating illuminated manuscripts: an effective tool for research and education. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 121–130, New York, NY, USA. ACM Press.

Agosti, M. and Melucci, M. (2000). Information retrieval techniques for the automatic construction of hypertext. In Kent, A. and Hall, C. M., editors, *Encyclopedia of Library and Information Science*, volume 66, pages 129–172. Marcel Dekker, New York, USA.

Agosti, M. and Smeaton, A. F., editors (1996). *Information Retrieval and Hypertext*. Kluwer Academic Publishers, Boston et al.

Amati, G. and van Rijsbergen, C. J. (2002). Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389.

Arms, W. Y. (2001). *Digital Libraries*. MIT Press, Cambridge, Mass.

Baader, F., Calvanese, D., McGuiness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook – Theory, Implementation and Applications*. Cambridge University Press, Cambridge, UK.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley.

Balog, K., Meij, E., and de Rijke, M. (2006). Language models for enterprise search: Query expansion and combination of evidence. In Voorhees and Buckland (2006).

Beitzel, S. M., Jensen, E. C., Cathey, R., Ma, L., Grossman, D. A., Frieder, O., Chowdhury, A., Pass, G., and Vandermolen, H. (2003). IIT at TREC 2003, task classification and document structure for known-item search. In Voorhees (2003), pages 311–320.

Belnap, N. (1977). A useful four-valued logic. In *Modern Uses of Multiple-Valued Logic*. Reidel, Dordrecht.

Berners-Lee, T., Fielding, R., and Masinter, L. (2005). Uniform Resource Identifier (URI): Generic syntax. Technical report, Internet RFC-3986.

Bernheim Brush, A. J. (2002). *Annotating Digital Documents for Asynchronous Collaboration*. PhD thesis, Department of Computer Science and Engineering, University of Washington.

Biebricher, P., Fuhr, N., Knorz, G., Lustig, G., and Schwantner, M. (1988). The automatic indexing system AIR/PHYS - from research to application. In *11th International Conference on Research and Development in Information Retrieval*, pages 333–342, Grenoble, France. Presses Universitaires de Grenoble.

Biron, P. V. and Malhotra, A. (2004). XML Schema part 2: Datatypes second edition. W3C recommendation 28 october 2004. Technical report, W3C. `http://www.w3.org/TR/xmlschema-2/`, last visited 14 September 2007.

Bookstein, A. and Swanson, D. R. (1974). Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*, 25:312–318.

Bottoni, P., Civica, R., Levialdi, S., Orso, L., Panizzi, E., and Trinchese, R. (2004). Madcow: a multimedia digital annotation system. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 55–62, New York, NY, USA. ACM Press.

Bottoni, P., Levialdi, S., and Rizzo, P. (2003). An analysis and case study of digital annotation. In Bianchi-Berthouze, N., editor, *Databases in Networked Information Systems, Third International Workshop, DNIS 2003, Aizu, Japan, September 22-24, 2003, Proceedings*, volume 2822 of *Lecture Notes in Computer Science*, pages 216–230. Springer.

Brocks, H., Stein, A., Thiel, U., Frommholz, I., and Dirsch-Weigand, A. (2002). How to incorporate collaborative discourse in cultural digital libraries. In *Proceedings of the ECAI 2002 Workshop on Semantic Authoring, Annotation & Knowledge Markup (SAAKM02)*, Lyon, France.

Buitelaar, P. and Declerck, T. (2003). Linguistic annotation for the semantic web. In Handschuh and Staab (2003a), pages 93–112.

Bush, V. (1945). As we may think. *Atlantic Monthly*. `http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm`.

Cabanac, G., Chevalier, M., Chrisment, C., and Julien, C. (2007). Collective annotation: Perspectives for information retrieval improvement. In *RIAO 2007: Proceedings of the 8th International Conference on Information Retrieval - Large-Scale Semantic Access to Content (Text, Image, Video and Sound)*, Pittsburgh, PA, USA. C.I.D. Paris, France.

Camon, E., Magrane, M., Barrell, D., Binns, D., Fleischmann, W., Kersey, P., Mulder, N., Oinn, T., Maslen, J., Cox, A., and Apweiler, R. (2003). The Gene Ontology Annotation (GOA) project: Implementation of GO in SWISS-PROT, TrEMBL, and InterPro. *Genome Res.*, 13:662–672.

Ceri, S., Gottlob, G., and Tanca, L. (1990). *Logic Programming and Databases*. Springer, Heidelberg et al.

Chiaramella, Y. and Kheirbek, A. (1996). An integrated model for hypermedia and information retrieval. In Agosti and Smeaton (1996), chapter 7.

Chiaramella, Y., Mulhem, P., and Fourel, F. (1996). A model for multimedia information retrieval. Technical report, FERMI ESPRIT BRA 8134, University of Glasgow.

Clauß, G. and Ebner, H. (1972). *Grundlagen der Statistik für Psychologen, Pädagogen und Soziologen*. Verlag Harri Deutsch, Frankfurt am Main und Zürich.

Constantopoulos, P. and Sølvberg, I. T., editors (2003). *Research and Advanced Technology for Digital Libraries. Proc. European Conference on Digital Libraries (ECDL 2003)*, Lecture Notes in Computer Science, Heidelberg et al. Springer.

Cooper, W. S., Gey, F. C., and Dabney, D. P. (1992). Probabilistic retrieval based on staged logistic regression. In Belkin, N. J., Ingwersen, P., and Pejtersen, A. M., editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in*

Information Retrieval. Copenhagen, Denmark, June 21-24, 1992, pages 198–210, New York. ACM.

Craswell, N., de Vries, A. P., and Soboroff, I. (2005). Overview of the TREC 2005 Enterprise Track. In Voorhees and Buckland (2005).

Craswell, N. and Hawking, D. (2003). Overview of the TREC-2003 web track. In Voorhees (2003).

Crestani, F. and Lalmas, M. (2001). Logic and uncertainty in information retrieval. In Agosti, M., Crestani, F., and Pasi, G., editors, *Lectures in Information Retrieval*, pages 179–206. Springer, Heidelberg et al.

Crestani, F., Lalmas, M., van Rijsbergen, C. J., and Campbell, I. (1998). "Is this document relevant?... probably": a survey of probabilistic models in information retrieval. *ACM Computing Surveys*, 30(4):528–552.

Croft, W. B., Moffat, A., van Rijsbergen, C. J., Wilkinson, R., and Zobel, J., editors (1998). *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York. ACM.

Croft, W. B. and Turtle, H. (1989). A retrieval model for incorporating hypertext links. In *Proceedings of the Hypertext '89*, pages 213–224, New York. ACM.

Del Bimbo, A., Gradmann, S., and Ioannidis, Y. (2004). Future research directions – 3rd DELOS brainstorming workshop report, DELOS Network of Excellence.

Delcambre, L. M. L., Maier, D., Bowers, S., Weaver, M., Deng, L., Gorman, P., Ash, J., Lavelle, M., and Lyman, J. (2001). Bundles in captivity: An application of superimposed information. In *Proceedings of the 17th International Conference on Data Engineering*, pages 111–120, Washington, DC, USA. IEEE Computer Society.

Denoue, L. and Vignollet, L. (2000). An annotation tool for web browsers and its applications to information retrieval. In *Proceedings of RIAO 2000, Paris, April 2000*.

Dowell, R. D., Jokerst, R. M., Day, A., Eddy, S. R., and Stein, L. (2001). The distributed annotation system. *BMC Bioinformatics*, 2(7).

Fagin, R. and Halpern, J. (1994). Reasoning about knowledge and probability. *Journal of the ACM*, 41(2):340–367.

Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, M. Y. (1995). *Reasoning about Knowledge*. MIT Press, Cambridge, Massachusetts.

Fraenkel, A. S. and Klein, S. T. (1999). Information retrieval from annotated texts. *Journal of the American Society for Information Science*, 50(10):845–854.

Frei, H. P. and Stieger, D. (1994). The use of semantic links in hypertext information retrieval. *Information Processing and Management*, 31(1):1–13.

Frisse, M. E. (1988). Searching for information in a hypertext medical handbook. *Communications of the ACM*, 31(7):880–886.

Frommholz, I. (2005a). Applying the annotation view on messages for discussion search. In Voorhees and Buckland (2005).

Frommholz, I. (2005b). What did the others say? Probabilistic indexing and retrieval models in annotation-based discussions. Bulletin of the IEEE Technical Committee on Digital Libraries. http://www.ieee-tcdl.org/Bulletin/v2n2/frommholz/frommholz.html.

Frommholz, I. (2007). Annotation-based document retrieval with probabilistic logics. In Fuhr, N., Kovacs, L., and Meghini, C., editors, *Research and Advanced Technology for Digital Libraries. Proc. of the 11th European Conference on Digital Libraries (ECDL 2007)*, Lecture Notes in Computer Science, pages 321–332, Heidelberg et al. Springer.

Frommholz, I., Brocks, H., Thiel, U., Neuhold, E., Iannone, L., Semeraro, G., Berardi, M., and Ceci, M. (2003). Document-centered collaboration for scholars in the humanities - the COLLATE system. In Constantopoulos and Sølvberg (2003), pages 434–445.

Frommholz, I. and Fuhr, N. (2006a). Evaluation of relevance and knowledge augmentation in discussion search. In Gonzalo, J., Thanos, C., Verdejo, M. F., and Carrasco, R. C., editors, *Research and Advanced Technology for Digital Libraries. Proc. of the 10th European Conference on Digital Libraries (ECDL 2006)*, Lecture Notes in Computer Science, pages 279–290, Heidelberg et al. Springer.

Frommholz, I. and Fuhr, N. (2006b). Probabilistic, object-oriented logics for annotation-based retrieval in digital libraries. In Nelson, M., Marshall, C., and Marchionini, G., editors, *Opening Information Horizons – Proc. of the 6th ACM/IEEE Joint Conference on Digital Libraries (JCDL 2006)*, pages 55–64, New York. ACM.

Frommholz, I., Knežević, P., Mehta, B., Niedereé, C., Risse, T., and Thiel, U. (2004a). Supporting information access in next generation digital library architectures. In Agosti, M., Schek, H.-J., and Türker, C., editors, *Digital Library Architectures: Peer-to-Peer, Grid, and Service-Orientation. Proceedings of the Sixth Thematic Workshop of the EU Network of Excellence DELOS*, pages 49–60, Cagliari, Italy.

Frommholz, I. and Larson, R. (2007). Report on the INEX 2006 heterogeneous collection track. *SIGIR Forum*, 41(1).

Frommholz, I. and Lechtenfeld, M. (2008). Determining the polarity of postings for discussion search. In *Proc. of the "Information Retrieval 2008" Workshop at LWA 2008, Würzburg, Germany.*

Frommholz, I., Thiel, U., and Kamps, T. (2004b). Annotation-based document retrieval with four-valued probabilistic datalog. In Rölleke, T. and de Vries, A. P., editors, *Proceedings of the first SIGIR Workshop on the Integration of Information Retrieval and Databases (WIRD'04)*, pages 31–38, Sheffield, UK.

Fuhr, N. (1992). Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255.

Fuhr, N. (2000). Probabilistic Datalog: Implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51(2):95–110.

Fuhr, N. and Buckley, C. (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9(3):223–248.

Fuhr, N. and Buckley, C. (1993). Optimizing document indexing and search term weighting based on probabilistic models. In Harman, D., editor, *The First Text REtrieval Conference (TREC-1)*, pages 89–100, Gaithersburg, Md. 20899. National Institute of Standards and Technology Special Publication 500-207.

Fuhr, N., Gövert, N., Kazai, G., and Lalmas, M. (2002). INEX: INitiative for the Evaluation of XML retrieval. In Baeza-Yates, R., Fuhr, N., and Maarek, Y. S., editors, *Proceedings of the SIGIR 2002 Workshop on XML and Information Retrieval.* `http://www.is.informatik.uni-duisburg.de/bib/docs/Fuhr_etal_02a.html`.

Fuhr, N., Gövert, N., Kazai, G., and Lalmas, M., editors (2003). *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the First INEX Workshop. Dagstuhl, Germany, December 8–11, 2002*, ERCIM Workshop Proceedings, Sophia Antipolis, France. ERCIM. `http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf`.

Fuhr, N., Gövert, N., and Rölleke, T. (1998). DOLORES: A system for logic-based retrieval of multimedia objects. In Croft et al. (1998), pages 257–265.

Fuhr, N. and Großjohann, K. (2004). XIRQL: An XML query language based on information retrieval concepts. *ACM Transactions on Information Systems*, 22:313–356.

Fuhr, N., Hansen, P., Mabe, M., Micsik, A., and Solvberg, I. (2001). Digital libraries: A generic classification and evaluation scheme. In *Proceedings European Conference on Digital Libraries*, pages 187–199, Heidelberg et al. Springer.

Fuhr, N., Lalmas, M., and Malik, S., editors (2004). *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop. Dagstuhl, Germany, December 15–17, 2003.* `http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf`.

Fuhr, N., Lalmas, M., Malik, S., and Kazai, G., editors (2006). *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005), Dagstuhl 28-30 November 2005, Lecture Notes in Computer Science*, volume 3977. Springer-Verlag GmbH.

Fuhr, N., Lalmas, M., Malik, S., and Szlavik, Z., editors (2005). *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers*, volume 3493. Springer-Verlag GmbH. http://www.springeronline.com/3-540-26166-4.

Fuhr, N., Lalmas, M., and Trotman, A., editors (2007). *Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006*, number 4518 in LNCS, Heidelberg et al. Springer.

Fuhr, N. and Pfeifer, U. (1991). Combining model-oriented and description-oriented approaches for probabilistic indexing. In *Proceedings of the Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 46–56, New York. ACM.

Fuhr, N. and Rölleke, T. (1997). A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems*, 14(1):32–66.

Fuhr, N. and Rölleke, T. (1998). HySpirit – a probabilistic inference engine for hypermedia retrieval in large databases. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT)*, pages 24–38, Heidelberg et al. Springer.

Furuta, R., Shipman, F. M., Marshall, C. C., Brenner, D., and Hsieh, H. (1997). Hypertext paths and the World Wide Web: Experiences with Walden's Path. In *Proc. of Hypertext '97: the Eighth ACM Conf. on Hypertext*, pages 167–176, UK. ACM.

Gertz, M., Sattler, K.-U., Gorin, F., Hogarth, M., and Stone, J. (2002). Annotating scientific images: a concept-based approach. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management, 2002*, pages 59–68.

Golovchinsky, G., Price, M. N., and Schilit, B. N. (1999). From reading to retrieval: freeform ink annotations as queries. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval*, pages 19–25, New York. ACM.

Gonçalves, M. A., Fox, E. A., Watson, L. T., and Kipp, N. A. (2004). Streams, structures, spaces, scenarios, societies (5s): A formal model for digital libraries. *ACM Trans. Inf. Syst.*, 22(2):270–312.

Güting, R. H. (1992). *Datenstrukturen und Algorithmen.* B.G. Teubner Stuttgart.

Halasz, F. G. (1988). Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Communications of the ACM*, 31(7).

Hammwöhner, R. and Thiel, U. (1987). Content oriented relations between text units – a structural model for hypertexts. In *HYPERTEXT '87: Proceeding of the ACM conference on Hypertext*, pages 155–174, New York, NY, USA. ACM.

Handschuh, S. and Staab, S., editors (2003a). *Annotation for the Semantic Web.* IOS Press.

Handschuh, S. and Staab, S. (2003b). Annotation of the shallow and the deep web. In Handschuh and Staab (2003a), pages 25–46.

Hawking, D., Vorhees, E., Craswell, N., and Bailey, P. (1999). Overview of the TREC-8 web track. In Voorhees and Harman (1999), pages 131–150.

Heery, R. and Lyon, L., editors (2004). *Research and Advanced Technology for Digital Libraries. Proc. European Conference on Digital Libraries (ECDL 2004)*, Lecture Notes in Computer Science, Heidelberg et al. Springer.

Hornby, A., Cowie, A., and Windsor Lewis, J., editors (1976). *Oxford Advanced Learner's Dictionary of Current English.* Oxford University Press.

International Organization for Standardization (1996). ISO/EIC 14977: Information technology – syntactic metalanguage – Extended BNF.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. In Nédellec, C. and Rouveirol, C., editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Heidelberg et al. Springer.

Kahan, J., Koivunen, M., Prud'Hommeaux, E., and Swick, R. (2001). Annotea: An open RDF infrastructure for shared web annotations. In *Proceedings of the WWW10 International Conference*, Hong Kong.

Kehoe, B. P. (1993). *Zen and the Art of the Internet. A Beginners Guide.* Prentice Hall, Englewood Cliffs, NJ, 2nd edition.

Klas, C.-P., Fuhr, N., and Schaefer, A. (2004a). Evaluating strategic support for information access in the DAFFODIL system. In Heery and Lyon (2004).

Klas, C.-P., Kriewel, S., and Schaefer, A. (2004b). Daffodil - Nutzerorientiertes Zugangssystem für heterogene digitale Bibliotheken. *dvs Band.*

Kleinberg, J. (1998). Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677.

Kouzes, R. T., Myers, J. D., and Wulf, W. A. (1996). Collaboratories: Doing science on the internet. *Computer*, 29(8):40–46.

Lalmas, M. and Rölleke, T. (2003). Four-valued knowledge augmentation for structured document retrieval. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 11(1):67–85.

Lechtenfeld, M. (2007). Sentiment Classification in Diskussionen. Master's thesis, Universität Duisburg-Essen, FB Ingenieurwissenschaften. In German.

Lucarella, D. and Zanzi, A. (1996). Information modelling and retrieval in hypermedia systems. In Agosti and Smeaton (1996), chapter 6.

Marshall, C. C. (1997). Annotation: From paper books to the digital library. In *Proceedings of the ACM Digital Libraries '97 Conference*, pages 131–140.

Marshall, C. C. (1998). Toward an ecology of hypertext annotation. In *Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space – structure in hypermedia systems*, pages 40–49.

Meghini, C., Sebastiani, F., Straccia, U., and Thanos, C. (1993). A model of information retrieval based on a terminological logic. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298–308, New York. ACM.

Mizzaro, S. (1998). How many relevances in information retrieval? *Interacting With Computers*, 10(3):303–320.

Müller, A. and Thiel, U. (1994). Query expansion in an abductive information retrieval system. In *In Proc. RIAO '94 - Intelligent Multimedia Information Retrieval Systems and Management*, pages 461–480.

Nagao, K. (2003). *Digital Content Annotation and Transcoding.* Artech House, Norwood (MA), USA.

Neuhold, E., Niedereé, C., Stewart, A., Frommholz, I., and Mehta, B. (2004). The role of context for information mediation in digital libraries. In Chen, Z., Chen, H., Miao, Q., Fu, Y., Fox, E., and Lim, E.-P., editors, *Proc. of the 7th International Conference on Asian Digital Libraries: International Collaboration and Cross-Fertilization (ICADL 2004)*, volume 3334 of *Lecture Notes in Computer Science (LNCS)*, pages 133–143, Heidelberg et al. Springer.

Nichols, D. M., Pemberton, D., Dalhoumi, S., Larouk, O., Belisle, C., and Twidale, M. B. (2000). DEBORA: Developing an interface to support collaboration in a digital library. In Borbinha, J. and Baker, T., editors, *Research and Advanced Technology for Digital Libraries. Proc. European Conference on Digital Libraries (ECDL 2000)*, volume 1923 of *Lecture Notes in Computer Science*, pages 239–248, Heidelberg et al. Springer.

Nilsson, N. J. (1986). Probabilistic logic. *Artificial Intelligence*, 28:71–87.

Nottelmann, H. and Fuhr, N. (2003). From uncertain inference to probability of relevance for advanced IR applications. In Sebastiani, F., editor, *25th European Conference on Information Retrieval Research (ECIR 2003)*, pages 235–250, Heidelberg et al. Springer.

Ogilvie, P. and Callan, J. (2004). Using language models for flat text queries in XML retrieval. In Fuhr et al. (2004), pages 12–18. `http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf`.

O'Reilly, T. (2005). What is Web 2.0 – design patterns and business models for the next generation of software. `http://www.oreilly.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`, last visited 2007-12-02.

Ovsiannikov, I. A., McNeill, T. H., and Arbib, M. A. (1999). Annotation technology. *Int. J. Hum.-Comput. Stud. 50*, 50(4):329–362.

Paepcke, A. (1996). Digital libraries: Searching is not enough–what we learned on-site. *D-Lib Magazine*, 2(5). `http://www.dlib.org/dlib/may96/stanford/05paepcke.html`.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proc. of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86.

Phelps, T. A. and Wilensky, R. (1997). Multivalent annotations. In Peters, C. and Thanos, C., editors, *Research and Advanced Technology for Digital Libraries. Proc. European Conference on Digital Libraries (ECDL 1997)*, volume 1324 of *Lecture Notes in Computer Science*, pages 287–303, Heidelberg et al. Springer.

Piwowarski, B., Faure, G.-E., and Gallinari, P. (2003). Bayesian networks and INEX. In Fuhr et al. (2003), pages 149–154. `http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf`.

Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In Croft et al. (1998), pages 275–281.

Robertson, S. E. (1977). The probability ranking principle in IR. *Journal of Documentation*, 33:294–304.

Robertson, S. E. and Sparck Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146.

Robertson, S. E. and Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In Croft, B. W. and van Rijsbergen, C. J., editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, London, et al. Springer-Verlag.

Robertson, S. E., Walker, S., Jones, S., and Hancock-Beaulieu, M. M. (1995). Okapi at TREC-3. In *Proceedings of the 3rd Text Retrieval Converence (TREC-3)*, pages 109–126, Springfield, Virginia, USA. NTIS.

Rölleke, T. (1998). *POOL: Probabilistic Object-Oriented Logical Representation and Retrieval of Complex Objects*. PhD thesis, University of Dortmund, Germany.

Rölleke, T. (2008). *Probabilistic Logical Models for Database (DB) and Information Retrieval (IR) – A Seamlessly Integrated DB+IR Technology*. In preparation.

Rölleke, T. and Fuhr, N. (1996). Retrieval of complex objects using a four-valued logic. In *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–214, New York. ACM.

Rölleke, T., Wu, H., Wang, J., and Azzam, H. (2007). Modelling retrieval models in a probabilistic relational algebra with a new operator: the relational Bayes. *The International Journal on Very Large Data Bases (VLDB)*, 17(1):5–37.

Ross, K. A. (1990). Modular stratification and magic sets for datalog programs with negation. In *PODS '90: Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 161–171, New York, NY, USA. ACM.

Röscheisen, M., Mogensen, C., and Winograd, T. (1995). Beyond browsing: shared comments, SOAPs, trails, and on-line communities. *Computer Networks and ISDN Systems*, 27(6):739–749.

Sanderson, M. and Zobel, J. (2005). Information retrieval system evaluation: effort, sensitivity and reliability. In Marchionini, G., Moffat, A., and Tait, J., editors, *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York. ACM.

Savoy, J. and Picard, J. (1999). Report on the TREC-8 experiment: Searching on the web and in distributed collections. In Voorhees and Harman (1999), pages 229–240.

Schenkel, R., Crecelius, T., Kacimi, M., Michel, S., Neumann, T., Parreira, J., and Weikum, G. (2008). Efficient top-k querying over social-tagging networks. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research & Development on Information Retrieval (SIGIR 2008)*, Singapore, Singapore. Accepted for publication.

Schilit, B. N., Golovchinsky, G., and Price, M. N. (1998a). Beyond paper: Supporting active reading with free form digital ink annotations. In *CHI 98 Conference Proceedings*, pages 249–256. ACM.

Schilit, B. N., Price, M. N., and Golovchinsky, G. (1998b). Digital library information appliances. In *Proceedings of the third ACM Conference on Digital libraries*, pages 217–226. ACM.

Searle, J. (1979). A taxonomy of illocutionary acts. In Searle, J., editor, *Expression and Meaning. Studies in the Theory of Speech Acts*, pages 1–29. Cambridge University Press, Cambridge.

Shipman, F., Price, M., Marshall, C. C., and Golovchinsky, G. (2003). Identifying useful passages in documents based on annotation patterns. In Constantopoulos and Sølvberg (2003), pages 101–112.

Storch, U. and Wiebe, H. (1989). *Lehrbuch der Mathematik. Band I: Analysis einer Veränderlichen*. BI Wissenschaftsverlag.

Thiel, U., Brocks, H., Frommholz, I., Dirsch-Weigand, A., Keiper, J., Stein, A., and Neuhold, E. (2004). COLLATE - a collaboratory supporting research on historic European films. *International Journal on Digital Libraries (IJDL)*, 4(1):8–12.

Trigg, R. H. (1983). *A network-based approach to text handling for the online scientific community*. PhD thesis, University of Maryland.

Turtle, H. and Croft, W. B. (1990). Inference networks for document retrieval. In *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 1–24, New York. ACM.

van Rijsbergen, C. J. (1986). A non-classical logic for information retrieval. *The Computer Journal*, 29(6):481–485.

van Rijsbergen, C. J. (1989). Towards an information logic. In *Proceedings of the Twelfth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 77–86, New York. ACM.

van Rijsbergen, C. J. (1992). Probabilistic retrieval revisited. *The Computer Journal*, 35(3):291–298.

Voorhees, E. M., editor (2003). *The Eleventh Text REtrieval Conference (TREC 2003)*, Gaithersburg, MD, USA. NIST.

Voorhees, E. M. and Buckland, L. P., editors (2005). *The Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, USA. NIST.

Voorhees, E. M. and Buckland, L. P., editors (2006). *The Fifteenth Text REtrieval Conference (TREC 2006)*, Gaithersburg, MD, USA. NIST.

Voorhees, E. M. and Harman, D., editors (1999). *The Eight Text REtrieval Conference (TREC-8)*, Gaithersburg, MD, USA. NIST.

Wolfe, J. L. (2000). Effects of annotations on student readers and writers. In *Proceedings of the fifth ACM conference on digital libraries*, pages 19–26. ACM.

Wong, S. K. M. and Yao, Y. Y. (1995). On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):38–68.

Xi, W., Lind, J., and Brill, E. (2004). Learning effective ranking functions for newsgroup search. In Järvelin, K., Allen, J., Bruza, P., and Sanderson, M., editors, *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 394–401, New York. ACM.