Information Retrieval: Concepts and Practical Considerations for Teaching a Rising Topic

Daniel Blank, Norbert Fuhr, Andreas Henrich, Thomas Mandl, Thomas Rölleke, Hinrich Schütze, Benno Stein

Abstract. The last two decades have seen an enormous increase in the amount of information available, in the form of text documents as well as multimedia data such as images, speech and video. As a result, information retrieval (IR) has become a central topic of computer science and related disciplines and is now part of many curricula for bachelor and master programs. In this article, we outline which concepts should be integral part of IR courses depending on the orientation of the degree program (e.g. business vs. research). In addition to the theoretical content of IR courses, we also address practical considerations, based on the authors' extensive experience in teaching IR. We comment on the suitability of a number of tools and systems and of different forms of teaching, including e-learning, in the IR classroom.

1 Motivation

Data volumes have been growing since computers were invented, and powerful database and information retrieval technologies have been developed to manage and retrieve large-scale data, to turn data into information. Since the mid 1990s, not only the data volume, but in particular the number of people exposed and dependent on information supply and search, has increased exponentially. Information (web) search has become an inherent and frequent part in the life of billions of people, and information search is important in both, professional *and* private contexts. Whereas before the mid 1990s, information search was a task mostly executed by trained and dedicated search professionals (librarians, database administrators), nowadays, professionals, semi-professionals, and hurried end-users share the same goal: to find relevant information quickly. Therefore, information retrieval (IR) is now part of various curricula for bachelor and master programs. These programs range from library science over information science to computer science; even programs in areas such as management science that used to regard IR as unimportant have now integrated this field as a key qualification.

Obviously, different target groups for teaching IR implicate different educational objectives. In the intended vocational field, IR systems might be *used*, *implemented*, *designed* or *managed*. These different objectives have to be considered when developing teaching concepts for IR.

Of course, there is a long way to go if we try to achieve a well established understanding of how to teach IR. Even the authors of this paper do not agree on all aspects considered in this paper. Nevertheless, the paper in hand stimulates the discussion. It is seen as a step and by no means as a final result. We hope that there will be a fruitful exchange of ideas in the future. We welcome comments on all opinions expressed in this article.

In the following we will address different aspects of teaching IR. Which topics should be part of the curriculum, and in which depth should these topics be addressed for the different target groups (section 2)? Since practical exercises are essential for learning IR, we need to address which tools and systems are useful in teaching IR (section 3). An important aspect is also the form of teaching. In addition to the standard classroom lectures, other forms we cover are tutorials, hands-on training, projects and seminars (section 4). Further important aspects of teaching IR are blended learning and e-learning concepts (section 5). Finally, section 6 concludes the paper.

2 Towards a Curriculum for IR

2.1 Contents

To compose a curriculum in IR, we merge suggestions from various text books (cf. section 2.2), synoptic articles such as [Croft, 1995] (Top 10 Research Issues), [Melucci and Hawking, 2006] (A perspective on Web Information Retrieval), or [Bawde et al., 2007] (Information Retrieval Curricula: Contexts and Perspectives), and IR summer schools. There seems to be a consensus that the main IR topics center around indexing (document/data analysis), ad-hoc retrieval, classification, interaction, and evaluation. On the background of library and information science Bawde et al. [Bawde et al., 2007] distinguish four related, but distinct subject areas: human information behaviour (HIB), information seeking (IS), information retrieval (IR), and general topics (Gen). Although the curriculum presented in [Bawde et al., 2007] has a strong focus on cognitive aspects it is useful for our considerations. Even a curriculum for computer scientists should not ignore these aspects. Nevertheless, a more system and implementation oriented approach will be better suited given our computer science background.

Before we discuss individual topics of the curriculum, we want to emphasise that in today's academic environment, the students in an IR course will have, depending on their study course, different motivations, expectations, and personal requirements. To simplify things a bit, we will differentiate the audience with respect to their expected working relationship to IR systems:

1. *IR system user* (U): For students falling in this category the efficient goal-oriented use of IR-systems is the main focus.

2. *Management* (M): In the expected future working context of students falling in this category there might be tasks regarding the supply of data and information in a company (e.g. knowledge management). Consequently, there is a business-oriented view on IR but with the need for a strong conceptual and technical background.

3. *Administration* (A): Here, the main focus is on the technical administration and optimisation of search tools.

4. *Development / Research* (D/R): The last group are students who would like to be part of research or implementation projects in the field of IR.

Table 1 gives an overview of our proposed curriculum. For the different target groups the appropriate depth of coverage is indicated. In the following we will discuss the different topic groups in greater detail.

2.1.1 Introduction

Although today everybody is using search engines, the roots and the background of IR need some explanation. To this end, different concrete search situations can be considered and first naive user experiments can be integrated into the concept.

In more detail a mission statement for IR should be given at first. A glimpse at the history of IR and its background in library science and information science should be given and important terms (e.g. *data*, *knowledge* and *information*) should be introduced. To communicate the various facets of IR, different usage scenarios can be discussed, starting from web search engines over search tasks in a digital library up to enterprise search scenarios or market investigation using IR techniques.

The knowledge of certain resources, the knowledge of necessary tools like thesauri, as well as the efficient use of such tools are sometimes the focus of entire courses. From a computer science perspective, awareness for professional search should be created and examples—maybe in a specific domain—should be presented. To this end, we have integrated the topics *search strategies* and *knowledge of resources* into the curriculum.

Finally—in order to sharpen the students' understanding—a discussion of the relationship between databases and IR should be given together with a consideration of the overlap (text extensions for relational databases, meta data search in IR systems, . . .).

2.1.2 IR Evaluation

The empirical evaluation of the performance of IR systems is of central importance because the quality of a system cannot be predicted based on its components. Since an IR system ultimately needs to support the user in fulfilling his information need, a holistic evaluation needs to set the satisfaction of the user and his or her work task as the yardstick. Such an evaluation is extremely difficult because it is influenced by many context factors such as previous knowledge of the user, his search skill and work environment. IR user studies typically provide test users with hypothetical search tasks in order to allow comparison. In such experiments, the user is asked to report his satisfaction with the system or its components. Most evaluations are system-oriented and follow the Cranfield-paradigm which will be presented below. A student should be aware of the different levels of evaluations that can be carried out, their potential results and disadvantages. If the curriculum also includes classes in human-computer interaction (HCI), the students might already have studied empirical evaluation and usability tests. That knowledge can be recalled in the class. Otherwise, it should be integrated into the IR class because it is not typically taught in other computer science classes. The

	U	M	A	D/R
Introduction				
Motivation and Overview	●	●	●	●
History of IR	●	●	●	●
Terms and Definitions	●	●	●	●
IR Topics and Usage Scenarios	●	●	●	●
Efficient Search: Search Strategies	●	○	○	○
Efficient Search: Knowledge of Resources	●	○	○	○
IR versus DB-driven Retrieval	○	●	○	●
IR Evaluation				
Performance Factors and Criteria	●	●	●	●
IR Performance Measures	○	●	●	●
Test Collections		○	●	●
System- vs. User-oriented Evaluation		●	○	●
Language Analysis				
Tokenisation		○	●	●
Filtering (stop words, stemming, etc.)	●	●	●	●
Meta Data	○	●	●	●
Natural Language Processing		○	○	●
Text- and Indexing Technology				
Pattern Matching	○	○	●	●
Inverted Files	○	○	●	●
Tree-based Data Structures		○	○	●
Hash-based Indexing		○	○	●
Managing Gigabytes	○	○	●	●
IR Models				
Boolean Model and its Extensions	●	●	●	●
Vector Space Model and its Generalisation	●	●	●	●
Probabilistic Retrieval		○	○	●
Logical Approach to IR		○	○	●
BM25 (Okapi)	○	●	●	●
Latent Variable Models (e.g. LSA)		○	○	●
Language Modelling		○	○	●
Cognitive Models and User Interfaces				
Information Seeking	●	●	●	●
Information Searching	●	●	●	●
Strategic Support	●	●	●	●
HCI Aspects	●	●	●	●
Input Modes and Visualisations	○	●	●	●
Agent-based and Mixed-initiative Interfaces	○	○	○	●
Data Mining and Machine Learning for IR				
Clustering	○	○	●	●
Classification	○	●	●	●
Mining of Heterogeneous Data	○	○	●	●
Special Topics (Application-oriented)				
Web Retrieval	○	●	●	●
Semantic Web	○	●	●	●
Multimedia Retrieval	○	○	●	●
Social Networks/Media		○	○	●
Opinion Mining and Sentiment Analysis		○	○	●
Geographic IR		○	○	●
Information Filtering	○	●	●	●
Question Answering	○	○	○	●
Special Topics (Technological)				
Cross-Language IR		○	○	●
Distributed IR	○	●	●	●
IR and Ranking in Databases		○	●	●
Learning to Rank		○	○	●
Summarisation		○	○	●
XML-Retrieval	○	●	●	●

Table 1: Topics for teaching IR together with their importance for different target groups (● = mandatory, ○ = overview only, blank = might be dispensable)

student should be at least aware of some of the difficulties involved in designing user experiments.

Evaluations based on the Cranfield-paradigm need to be the main focus of a lecture on evaluation in IR. Research has adopted this evaluation scheme which tries to ignore subjective differences between users in order to be able to compare systems and algorithms. The user is replaced by a prototypical and constant user. Relevance judgments are provided out by domain experts who evaluate the relevance of a document independent of subjective influences [Buckley and Voorhees, 2005]. In a lab class, students could experience the subjectivity of relevance judgments in an exercise.

The most important measures based on the relevance judgments are recall and precision. Recall shows how good a system is in finding relevant documents whereas precision measures how good a system is in finding only relevant documents without ballast. Many different evaluation measures have been suggested. The basic objectives for e.g. binary preference (bpref) and cumulative gain measures should be mentioned in a lecture. In a lab class, students could experiment with different measures to see whether they lead to different results.

Students need to know the main evaluation initiatives and should know some typical results. The three major evaluation initiatives are historically connected. The Text REtrieval Conference (TREC)[1] was the first large effort which started in 1992 [Buckley and Voorhees, 2005]. Subsequently, the Cross-Language Evaluation Forum (CLEF)[2] and the NII Test Collection for IR Systems (NTCIR)[3] adopted the TREC methodology and developed specific tasks for multilingual and cross-lingual searches. TREC achieved a high level of comparability of system evaluations for the first time in information science [Robertson, 2008]. The test data and collections have stimulated research and are still a valuable resource for development. The initial TREC collections for ad-hoc retrieval were newspaper and newswire articles. In the first few years, the effectiveness of the systems approximately doubled. In order to cope with the new requirements and the changing necessities of different domains and information needs, new tasks were continuously established [Mandl, 2008]. Evaluation initiatives provide collections of documents, topics as descriptions of information needs and after the experiments of the participating research groups, they organise the intellectual relevance assessments and publish comparative results.

2.1.3 Language Analysis

Traditionally, IR takes a rather simple approach to compositional semantics: under most IR models the interpretation of a document is based on the (multi) set of the words it contains. I.e., such so-called bag-of-word models ignore the grammatical concepts that govern sentence construction and text composition [Jurafsky and Martin, 2008].

The first step in IR language analysis is tokenisation, where the raw character stream of a document is transformed into a stream of units, which will be used as terms later on. The subsequent steps can be grouped into two categories: (a) term normalisation and (b) term selection. The first aims at the formation of term equivalence classes and includes case-folding, expanding of abbreviations, word conflation (i.e. stemming), and normalisation of dates and numbers. Term selection, on the other hand, aims at extracting the content carrying words from a unit of text. Both term normalisation and term selection are language dependent.

Highly frequent and uniformly distributed terms such as stop words (e.g. 'the', 'a', 'and') are not well suited to discriminate between relevant and non-relevant documents, assuming topical similarity. Hence these terms are usually removed. Note, however, that for the analysis of a document's genre, sentiment, or authorship, stop words play an important role. Other forms of term selection include collocation analysis and noun phrase or key phrase extraction. The problem of word sense disambiguation is addressed differently by the different IR retrieval models; technologies include latent semantic analysis, synonym sets expansion, collocation analysis, and automated or manual tagging.

Natural language processing, NLP for short, is a large research field on its own [Manning and Schütze, 2000]. Currently, the application of NLP techniques in IR is limited to shallow NLP techniques (e.g. part-of-speech analysis); however, from a technological viewpoint IR and NLP are growing together. The driving force behind this process is threefold: the need to employ more elaborate NLP techniques for advanced information retrieval tasks such as plagiarism analysis, fact retrieval, or opinion mining, the increased computing power, the recent advances in NLP, owing to the use of machine learning techniques.

2.1.4 Text- and Indexing Technology

From a computer science point of view this field is the most traditional one. It covers technology for pattern matching, efficient data storage, hashing, and text compression.

Patterns can be of different types, ranging from simple to complex: terms, substrings, prefixes, regular expressions, patterns that employ a fuzzy or error-tolerant similarity measure. Consider the phonological similarity between two words as an example for a tolerant measure. Technology for pattern matching comprises the classical string searching algorithms (Knuth-Morris-Pratt, Boyer-Moore-Horspool, Karp-Rabin), heuristic search algorithms, but does also require sophisticated data structures, such as an n-gram inverted index, suffix trees and suffix arrays, signature files, or tries (Patricia in particular).

The central data structure for efficient document retrieval from a large document collection is the inverted file. Basically, an inverted file associates the terms in a dictionary with the respective term occurrences in the documents. Specialised variants and advanced improvements exploit certain retrieval constraints and optimisation potential—for example: memory size, distribution of queries, proximity and co-occurrence queries, knowledge about the update frequency of a collection (static versus dynamic), presorted occurrence lists, meta indexes and caching strategies [Witten et al., 1999].

Another retrieval technology is hashing. One distinguishes between exact hashing, which is applied for exact search (e.g. with MD5), and fuzzy hashing, also called hash-based similarity search: two document vectors are considered as similar if they are mapped on the same hash key. I.e., hashing reduces a continuous similarity relation to the binary concept "similar or not similar". Similarity hashing is applied for near similarity search in large collections, near-duplicate detection, and plagiarism analysis. Fuzzy hashing is an incomplete technology, whereas the

[1]`http://trec.nist.gov/`, last visit: 19.1.09
[2]`http://www.clef-campaign.org/`, last visit: 19.1.09
[3]`http://research.nii.ac.jp/ntcir/`, last visit: 19.1.09

tradeoff between precision and recall can be controlled to some extent [Stein, 2007].

Text compression is employed to reduce the memory footprint of index components, or to alleviate the bottleneck situation when loading large occurrence lists into the main memory.

2.1.5 IR Models

IR models can be viewed as—mostly mathematical—frameworks to define scores of documents. The scores allow to rank documents, and the ranking is expected to reflect the notion of relevance, that is relevant documents should have high scores while non-relevant documents should have low scores.

Ranking is nowadays standard, whereas the first retrieval model, namely the Boolean model, did not provide ranking. Models such as coordination level match (count the terms that are in both document and query), extended Boolean (weighting of query terms), and fuzzy retrieval helped to add ranking to Boolean expressions. The Boolean AND allows to restrict the answer set, but by adding constraints, relevant documents might be suppressed, just because they do not satisfy one criterion. Too specific (that is conjunctive) queries lead to what is referred to as the "empty-answer problem", whereas too broad (that is disjunctive) queries lead to the so-called "many-answer problem".

A main breakthrough for retrieval was the usage of vector-space algebra, leading to what is referred to as the vector-space model (VSM, promoted by the SMART system, [Salton et al., 1975]). The VSM views documents and queries as vectors, and the similarity of vectors (usually the angle between vectors) defines the score. The vector components correspond to document features, in particular to the terms of the vocabulary considered. The TF-IDF (term frequency (TF) times inverse document frequency (IDF)) was developed to form the vector components: with TF being a measure to be high for terms that are *frequent* within the document, and IDF being a measure to be high for terms that are *rare* in the whole collection, the VSM delivers a retrieval quality that—until today—is a strong baseline when evaluating IR systems.

The 1970s saw the development of what became known as the probabilistic retrieval model, or, more precisely the binary independence retrieval model [Robertson and Sparck Jones, 1976]. Foundations such as the probability of relevance and the probabilistic ranking principle were developed, and form the basis of today's probabilistic models.

The 1980s brought the logical approach to IR. The probability of a logical implication between document and query is viewed to constitute the score. This "model" is mainly theoretical. It is useful to explain other IR models [Wong and Yao, 1995], and to define probabilistic logics for executing IR models on databases.

The 1990s brought the retrieval model BM25 [Robertson et al., 1994]. BM25 (best match version 25) can be viewed as a successful mix of TF-IDF, binary independence retrieval, and document length normalisation (the so-called pivoted document length normalisation). Also, theoretically, BM25 is motivated by the 2-Poisson model, an application of the general Poisson probability to IR.

The late 1990s saw the paradigm of language modelling (LM) to be used in IR, where language modelling is a probabilistic retrieval model [Croft and Lafferty, 2003]. With some respect, LM is more probabilistic than the previously mentioned BIR model.

The theory and contributions of IR models are covered in extensive literature background among which are [Rijsbergen, 1979] (only online), [Wong and Yao, 1995] (logical framework to explain IR models), [Grossman and Frieder, 2004] (text book), [Rölleke et al., 2006] (matrix framework to explain IR models), [Belew, 2000] (text book), and [Robertson, 2004] (understanding IDF).

2.1.6 Cognitive models and user interfaces

Whereas database systems are mostly accessed from application programs, queries to IR systems are typically entered via a user interface. Thus, in order to achieve a high retrieval quality for the end user, cognitive aspects of interactive information access as well as the related problems of human-computer interaction have to be addressed.

Cognitive IR models distinguish between information seeking and searching. The former regards all activities related to information acquisition, starting from the point where the user becomes aware of an information need, until the information is found and can be applied. Popular models in this area have been developed by Ellis [Ellis, 1989] and Kuhltau [Kuhlthau, 1988]. In contrast, information searching focuses only on the interaction of the user with an information system. Starting from Belkin's concept of "Anomalous state of knowledge" [Belkin, 1980] or Ingwersen's cognitive model [Ingwersen, 1992] regarding the broad context of the search, more specific approaches include the berrypicking model [Bates, 1989], the concept of polyrepresentation or Belkin's episodic model. In all these models, the classical view of a static information need is replaced by a more dynamic view of interaction. For guiding the user in the search process an IR system should provide strategic support; for this purpose, Bates [Bates, 1990] identified four levels of search activities that are applied by experienced searchers, for which a concrete system can provide different degrees of system support.

The design of the *user interface* to an IR system also is a crucial topic [Hearst, 1999]. First, HCI aspects like Shneiderman's design principles [Shneiderman, 1998] and interaction styles should be introduced. Classical input interfaces include command languages, forms and menus. A large number of visualisations for IR have been developed [Hearst, 1999, Mann, 2002], either as static views or allowing for direct manipulation. In order to free the user from routine tasks in search, agent-based interfaces [Lieberman, 1995, Shneiderman and Maes, 1997] have been proposed, but more recent developments favor mixed-initiative interfaces [Schaefer et al., 2005].

2.1.7 Data Mining and Machine Learning for IR

Classification methods and data mining techniques like clustering—which we will jointly refer to as "machine learning"—were originally a neglected part of the information retrieval curriculum. However, in recent years the importance of machine learning for information retrieval has increased significantly, both in research and in practical IR systems. This is partly due to the fact that documents are closely integrated with other data types, in particular with links and clicks on the web; and exploiting data types such as links and clicks often necessitates the use of machine learning. Closely connected to the heterogeneity of data types in large IR systems is the fact that documents in today's typical collections are extremely diverse in quality and origin. Classification is often needed to classify documents according to their expected utility to the user. Spam detection is perhaps the most important example for this. Finally, many recent improvements in core information retrieval have come from classification and clustering, e.g. viewing document retrieval as a text classification problem [Manning et al., 2008, chapters 11 & 12] or improving retrieval performance using clustering [Liu and Croft, 2004].

These uses of machine learning in information retrieval theory and applications should guide the selection of machine learning topics for information retrieval courses. Machine learning methods that are frequently used for classifying documents in the context of IR include Naive Bayes, Rocchio, and Support Vector Machines (SVMs). All three are efficient enough to be able to scale up to the large document collections that are typical of the internet age.

For clustering, the classical hierarchical clustering methods such as single-link and complete-link clustering offer students who are new to the subject easy access to the basic ideas and problems of clustering. It is important to present clustering in the context of its applications in IR such as search results clustering [Manning et al., 2008, ch. 16] and news clustering[4] because it is sometimes not immediately obvious to students how clustering contributes to the core goal of information finding.

If there is room for a data mining technique other than clustering, then PageRank [Brin and Page, 1998] is a good choice since it exemplifies the interaction of textual documents with complex meta data such as links and clicks. In our experience, students show great interest in PageRank and related link analysis algorithms—not least because they have personal experience with the web and would like to understand how the search engines they use every day rank documents.

Much work in machine learning requires a deeper knowledge of mathematical foundations in analysis and algebra than most computer science students have. It is therefore important to avoid machine learning methods that are beyond the capabilities of most students. Naive Bayes, Rocchio, hierarchical clustering and PageRank are examples of algorithms that all computer science students should be able to understand and are therefore good choices for an IR course.

2.1.8 Special Topics

There are many active research fields in information retrieval. Some of them are already of great commercial importance and others will have to show their potential in the future or have found their niche. One indication for which topics are currently hot is given by the sessions and workshops organised at the bigger IR conferences such as the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval [Myaeng et al., 2008] or the European Conference on IR Research [Macdonald et al., 2008]. Another indication might be seen in the evaluation tracks considered at TREC, CLEF, or the INitiative for the Evaluation of XML-Retrieval (INEX)[5].

In table 1 a selection of topics is given together with a rough assessment of their importance for the target groups. In our perception even IR users at an academic level should be aware of web search topics such as the PageRank algorithm, problems of crawling or the basics of search engine optimisation. Semantic web technology [Shadbolt et al., 2006], multimedia objects, and structured documents—especially XML documents—have had a strong influence on IR research and basic knowledge in these areas will be important to assess innovations in IR in the next years. Since IR systems themselves and the collections they have to cover are becoming more and more distributed a basic understanding of related aspects such as source selection strategies or schema integration methods seems essential. Furthermore, we have added *question answering* and *information filtering* to the topics which should be covered at least cursory for IR users, because they represent specialised perspectives demonstrating the broader applicability of IR techniques in special usage scenarios.

Other topics, such as *social media IR*, *cross language IR*, *geographic IR*, or *opinion mining* might also be of interest to IR users, but seem more dispensable for this target group if there is not enough time to cover these topics.

2.2 Literature and Forms of Teaching

The more stable aspects of the topics listed in table 1 are covered in IR textbooks that are available in English [Grossman and Frieder, 2004, Baeza-Yates and Ribeiro-Neto, 1999, Manning et al., 2008, Croft et al., 2009] as well as in German [Ferber, 2003, Stock, 2007, Henrich, 2008]. The more advanced topics currently discussed in research are addressed in IR conferences and journals such as SIGIR [Myaeng et al., 2008] or ECIR [Macdonald et al., 2008].

It has to be mentioned, that different forms of presentation are applicable when teaching IR. Besides lectures there are tutorials, lab classes with hands-on-training (usually performed on one's own) and projects (usually performed in groups). We will discuss the latter three in section 4. In this section we discuss two different forms of lectures.

First of all, there is the *classical lecture* with the professor giving a talk and trying to engage students by interspersing questions and short discussions. Obviously, the extent to which meaningful interaction is possible depends on the number of students in the class.

Another concept is the *reading club or seminar-style class*. Here chapters of a book, research papers, or research topics are given to the students. The students have to work through these topics till the next meeting and then the contents are discussed. Obviously, this concept is more appropriate for small groups and advanced topics. However, in such situations the dialog-oriented style of a reading club can motivate the students and foster autonomous learning.

2.3 Packages and Levels

One problem with curricular considerations is that in the end, a course or a group of courses has to fit into the framework of bachelor or master programs. In this context the available workload is usually predefined—in Europe frequently measured in ECTS (European Credit Transfer and Accumulation System) credit points. Assuming that one ECTS credit point relates to a workload of 30 hours for the average student, a group of comprehensive IR modules including lectures, exercises and projects

[4] See, e.g. `http://news.google.com/`, last visit: 19.1.09
[5] `http://www.inex.otago.ac.nz/about.html`, last visit: 19.1.09

could easily comprise 20 or more ECTS credits. However, in many programs only a smaller portion will be available.

Another problem comes from the fact that at least three types of students have to be distinguished. There are bachelor and master students in programs where IR should be part of the core curriculum. Such programs will usually be computer science, applied computer science or information science programs. Obviously, there should be courses for both groups and therefore in many cases there will be the need for an IR course for bachelor students and an (advanced) IR course for master students. With respect to the topics listed in table 1 a course for bachelor students could for example be restricted to the extent indicated for "IR system users" in the left column. If considered useful, basic implementation techniques and additional IR models can be added if the available ECTS credit points permit. In any case, exercises and small projects should be included already in bachelor level courses to facilitate the learning success. For master students the remaining topics together with more comprehensive projects can be offered.

Finally, there is a growing need to provide IR courses as a secondary subject for students in more loosely related programs. In fact, basic information retrieval competence can be seen as a domain-spanning key qualification. If enough teaching capacity is available and the potential audience is big enough, specialised courses for IR as a secondary subject can be beneficial in this respect, because otherwise there is the danger that the expectations of the students and the previous knowledge are too diverse. If computer science students and students learning IR as a secondary subject participate in the same course some students might be bored and others overchallenged. On the other hand, one could argue that such a mixed audience is beneficial for the students, since it is a good preparation for working in interdisciplinary teams. Although this argument has some truth, the challenge for the lecturer is high.

To sum up, the decision whether there should be one joint IR course or different IR courses for bachelor students in computer science (CS) or information science programs, on the one hand, and students studying IR as secondary subject, depends on the local parameters (teaching capacity, number of students etc.). A compromise in this respect might be to design a series of courses for different target groups as depicted in table 2.

	IR as secondary subject	CS Bachelor	CS Master
IR *A*	●	●	
IR *B*		●	
IR *C*			●

Table 2: Possible breakdown of IR courses

3 IR Systems and Tools for Teaching

In the following we will describe IR systems and tools that can be utilised when teaching IR. Systems and tools relevant to teaching IR vary from small single purpose demonstrators to full blown IR systems. Our analysis is twofold. First, we present different types of systems that can be used out-of-the-box without any need for tuning or adapting the source code (section 3.1). Some of the systems are characterised by a commercial background, others have appeared as prototypes developed by universities. Second, we present IR systems and tools that can be applied when developing IR applications (section 3.2). A main characteristic of these systems is that the software is open-source.

3.1 IR systems to show/use

Analysing the behaviour of IR systems in a structured way and getting to know best practices may be a beneficial task for IR students. By fulfilling typical search tasks, students can compare different systems, e.g. their graphical user interface and the way how results are presented as well as the performance of the systems by applying typical IR performance measures. Additionally, the systems can be shown during lectures in order to motivate or clarify concepts that are explained theoretically.

- *Web search engines*: Popular web search engines such as Google[6] or Live[7] are probably the most popular IR systems on the web. Amongst others, web search engines can be used to motivate IR research. Web search engines offer a ranking of search results that can be analysed by students reflecting ranking algorithms such as Google's PageRank [Brin and Page, 1998]. Applications such as Clusty[8] apply document clustering techniques on the search results to (re-)structure the result set. An example for a search engine providing relevance feedback facilities is scour[9]. Question answering is for example provided by Lexxe[10].

- *Web catalogues*: In contrast to typical web search engines, web catalogues such as DMOZ[11] or Yahoo[12] classify web pages according to different topics such as *sports*, *finance* or *travel*.

- *Tagging systems*: The collaborative annotation of large document collections has become popular in the last years. Delicious[13] for example allows for collaborative bookmarking. Users can share their bookmarks and annotate them with keywords in order to make them searchable. Flickr[14] allows for the tagging of images that users can upload. Tagging systems in general are a good basis for students to explore typical aspects of IR (vagueness of language, the need for cross-language IR, etc.).

- *Digital libraries*: Many universities offer their students free access to digital libraries such as IEEE Explore[15] or ACM digital library[16]. Digital libraries give students an impression of how to make document collections searchable that are restricted to certain domains. Information that should be indexed (author, conference, etc.) as well as different techniques for searching (Boolean retrieval, faceted search, etc.) can be identified. An example for a user-oriented access system for digital libraries is DAFFODIL[17] (Distributed Agents

[6]`http://www.google.com`, last visit: 19.1.09
[7]`http://www.live.com`, last visit: 19.1.09
[8]`http://clusty.com/`, last visit: 19.1.09
[9]`http://www.scour.com`, last visit: 2.2.09
[10]`http://www.lexxe.com/`, last visit: 2.2.09
[11]`http://www.dmoz.org/`, last visit: 19.1.09
[12]`http://www.yahoo.com/`, last visit: 19.1.09
[13]`http://www.delicious.com/`, last visit: 19.1.09
[14]`http://www.flickr.com/`, last visit: 19.1.09
[15]`http://ieeexplore.ieee.org/`, last visit: 19.1.09
[16]`http://portal.acm.org/dl.cfm`, last visit: 19.1.09
[17]`http://www.daffodil.de/`, last visit: 2.2.09

for User-Friendly Access of Digital Libraries). If the students' major subject is not in computer science, digital libraries addressing their major subject should be used (examples are: vascoda[18], STN[19], and DEPATIS[20]).

- *Prototypes for other search techniques*: Traditional Boolean retrieval can mostly be explored using online search facilities of local libraries. More comprehensive techniques such as faceted search/browsing can be studied analysing publicly available research prototypes such as Flamenco[21].

- *Applets & animations*: Many algorithms used in different areas of IR are visualised on the web. Applets and animations can be inspected by the students in order to support learning; examples can be found in the *Teaching IR* subtree on the web site of FG-IR[22].

- *Programmable IR tools*: Terrier[23] and MG4J[24] are IR systems with an academic background. They offer basic capabilities such as stop word removal and stemming. Initially developed in Java[25] as open-source software, they can be parametrised directly for the indexing and searching of document collections. Various IR models are provided and can be explored without a need to modify the sources. Originally, Terrier was designed to support web search research. It has since been extended to support desktop and intranet search.

  Rapidminer[26] is a data mining tool which offers various features that can also be used in teaching IR. As a rich user interface is provided, typical IR tasks can be performed without any programming skills. The extraction of document representations from different input formats is supported by various operators allowing e.g. for stop word removal or stemming. Based on extracted document representations, tasks such as clustering or text classification can be performed. A large number of clustering algorithms (e.g. *k*-Means) and classification techniques (e.g. based on support vector machines) can be used. Rapidminer integrates and extends the well-known machine learning library WEKA[27].

  Of course, it is also possible to use/adapt these systems by altering the source code. Therefore, they could as well be classified into the group of applications presented in section 3.2.

3.2 IR frameworks and libraries to adapt

The open-source IR systems and libraries that we present in the following are only a small selection of available tools. We mainly restrict this selection to software written in Java as Java has become more and more popular and many universities teach it in the computer science curriculum. Nevertheless, many IR tools are implemented in programming languages such as C/C++ or Perl and Unix tools provide useful functionality to realise IR systems [Riggs, 2002]. As a consequence, IR courses designed for IR system developers should address the implementation of Unix-based IR systems as well.

Middleton and Baeza-Yates [Middleton and Baeza-Yates, 2007] give a more detailed overview and compare 17 search engines after having eliminated some outdated projects and those that are no longer maintained.

Apache Lucene[28] is an open-source IR library. It is provided under the Apache software licence and can therefore be used in commercial products. Originally, Lucene was written in Java. It has been ported to a number of other programming languages, including C#, C++, Python and Perl. Lucene covers aspects of indexing and querying. Tasks regarding result presentation and crawling are not supported by Lucene. Lucene is well documented and a number of textbooks are available that can support students in getting a comprehensive introduction into Lucene (e.g. [Hatcher and Gospodnetic, 2004]). In the context of Lucene a couple of programming libraries have appeared that are based on Lucene and offer additional functionality.

Apache Solr[29] extends Lucene providing a search server. Solr's features include the XML/HTTP and JSON interfaces, hit highlighting capabilities, support for faceted search, caching, replication, and a web-based administration interface.

Apache Nutch[30] is also based on Lucene. Lucene is hereby extended with typical features of a web search engine. A crawler is provided within Nutch that supports the gathering and analysis of web pages. Based on the crawled and indexed web pages a link graph can be extracted and administered within Nutch. Since version 0.8 Nutch supports the Hadoop architecture. Hadoop[31] implements a distributed file system as well as Google's MapReduce algorithm [Dean and Ghemawat, 2004] that supports the processing of large amounts of data in a distributed environment.

Core retrieval tasks are supported by libraries such as Terrier, MG4J, Lucene and its extensions. Furthermore, libraries also exist that support various comprehensive tasks addressed in the curriculum. A list of tools can be found on the web site of FG-IR[32].

4 Tutorials, Exercises and IR Projects

A number of tasks can be addressed when teaching IR skills in tutorials, exercises and small projects:

- *Using retrieval systems to find documents relevant for given information needs*: Such exercises can help students understand why search is a hard problem and what typical capabilities of today's search systems are. Freely accessible tools (described in section 3.1) can be used in order to design the exercises.

- *Evaluating and comparing the quality of retrieval results achieved by different IR systems*: Performance analysis of IR systems is an important aspect of the IR curriculum. In order to gain experience in calculating typical measures such as recall and precision, the analysis of a small number of web search engines might be interesting. Given a certain

[18]`http://www.vascoda.de/`, last visit: 2.2.09
[19]`http://www.stn-international.com/stn_sneak.html`, last visit: 2.2.09
[20]`http://depatisnet.dpma.de`, last visit: 2.2.09
[21]`http://flamenco.berkeley.edu/`, last visit: 19.1.09
[22]`http://www.fg-ir.de`, last visit: 19.1.09
[23]`http://ir.dcs.gla.ac.uk/terrier/`, last visit: 19.1.09
[24]`http://mg4j.dsi.unimi.it/`, last visit: 19.1.09
[25]`http://java.sun.com/`, last visit: 19.1.09
[26]`http://www.rapidminer.com`, last visit: 19.1.09
[27]`http://www.cs.waikato.ac.nz/ml/weka/`, last visit: 19.1.09
[28]`http://lucene.apache.org/`, last visit: 19.1.09
[29]`http://lucene.apache.org/solr/`, last visit: 19.1.09
[30]`http://lucene.apache.org/nutch/`, last visit: 19.1.09
[31]`http://hadoop.apache.org/core/`, last visit: 19.1.09
[32]`http://www.fg-ir.de`, last visit: 19.1.09

information need, students can use the search engines and compare their performance by calculating typical IR performance measures.

Another interesting experience might be to examine different types of query formulation and their consequences for the retrieval. For example, students may benefit from trying different modes of querying on image search engines: query by sketch, query by example and search for images with particular textual annotations.

- *Applying algorithms and formulas manually*: There is a rich set of fundamental IR algorithms that can be applied manually in order to understand the algorithms in detail. Some examples are the PageRank algorithm [Brin and Page, 1998], the algorithm of Buckley and Lewit for determining the *k* most similar documents when applying the vector space model [Buckley and Lewit, 1985], or inserting and querying signature trees [Deppisch, 1986].

  Besides basic algorithms, IR models are well suited for performing basic calculations manually. Document representations for a small set of sample documents can be computed and afterwards documents can be matched against sample queries manually.

- *Implementing IR algorithms*: Of course, implementing some of the already mentioned algorithms is also promising. Small source skeletons can aid students in focusing only on the critical aspects of the algorithms avoiding tedious programming.

- *Reading exercises*: Especially in a master course, students are encouraged to gain some insights into research. Therefore, reading classical IR papers (e.g. from [Sparck Jones and Willett, 1997, Moffat et al., 2005]) or selected papers from recent conferences is a beneficial experience. Extracting the key aspects of the papers might be a task in an exercise. Alternatively, students can apply models or perform calculations that are suggested in the papers. Although small examples do not always show the true benefits of the presented approaches, they give some insights and leverage the burden of understanding the model/approach.

Having focused on more fine-grained tutorials and exercises in this section so far, we will now briefly describe three possible IR programming projects. These are just three basic examples amongst various other topics for IR projects.

- *Implementing a basic IR framework from scratch*: Within this project a small IR framework is implemented using only the Java Platform, Standard Edition (J2SE) without applying any of the IR libraries and frameworks described in section 3.2. The project is well suited for a bachelor course in IR. Basic Java programming skills as well as a course on algorithms and data structures are compulsory.

  At the beginning of the course students implement a recursive directory crawler. Later, a tokeniser is developed. In three filtering steps, case-folding, stop word removal and stemming is applied (using e.g. the Porter stemmer[33]). In an elegant way, tokenising and the three filtering steps can be refactored w.r.t. the pipes and filter pattern presented for example in [Buschmann et al., 1996].

  An inverted file can be constructed using Java's built-in data structures. Afterwards, in order to support Boolean retrieval, union and intersection of posting lists are implemented. In a second branch, document representations based on TF-IDF are computed. Various optimisations of the inverted file are possible, e.g. swapping document references, sorting the files by document ID. Finally, the algorithm of Buckley and Lewit for determining the *k* most similar documents [Buckley and Lewit, 1985] can be implemented.

  All programming tasks are extensively explained in short briefings at the beginning of a session. Students can work in teams. If there is additional time, the framework can be extended in many directions, e.g. integrating web crawling facilities, designing a user interface, evaluating the system, or parsing different document types. The latter is also a major concern in the following project.

  It has to be noted that the educational objective of this project is to deepen the students' understanding of basic IR algorithms. The students may misunderstand the implementation of such basic algorithms and become reluctant to using tools and libraries. Therefore, the lecturer should clarify this aspect.

- *Implementing desktop search using frameworks and libraries*: Here we use Lucene, briefly described in section 3.2, and various other libraries in order to design a small desktop search engine for full-text indexing of personal document collections. Another possibility would be to devise a project concerned with the design of a prototypical web search engine [Cacheda et al., 2008].

  This project is designed to implement a small desktop search application that indexes different document formats. At the beginning of the project the basics of Lucene are explained to the students. Key concepts such as *analysis*, *documents* and *fields* are emphasised. In a first step, students index their local file system with the help of a file crawler that traverses the file system. Afterwards, libraries for extracting the content of different document types are employed in order to index this information. By analysing the corresponding APIs students implement extraction mechanisms for different file types and index the content of the files with Lucene. Luke[34], a tool for inspecting Lucene's index can directly be employed analysing the consequences of tokenising and filtering. Basic query formulation is also possible with the help of Luke.

  To address language specific requirements, students can implement/apply their own analysing mechanisms such as tokenisers and filters for German. With the help of Luke, students always get immediate feedback about the consequences of their changes.

  After having introduced the basic properties of Lucene's query engine (query syntax, document scoring, . . .), students are asked to implement query processing. All tasks are introduced in a fine-grained way. There are many possibilities to extend this project: designing a user interface, extending the

[33]`http://tartarus.org/~martin/PorterStemmer/`, last visit: 19.1.09

[34]`http://www.getopt.org/luke/`, last visit: 19.1.09

framework with a web crawler, including linguistic analysis, etc.

- *Design and development of a (small) web search engine in a Unix environment*: This project covers the aspects of IR from data analysis over indexing to retrieval and evaluation. Students build a tokeniser to analyse some web pages (can be easily gathered via `wget` Unix command). Then, the collection is indexed, and the students prepare a layer that receives queries and returns results and result pages (page construction, snippet generation). The project involves the development of a basic GUI (query input, result browsing). This project trains the IR and software engineering skills of students, and the motivation is to "beat" a favourite web search engine for selected queries. Unix tools form a powerful basis for such a project [Riggs, 2002].

5 E-Learning for IR

Today, teaching and learning are generally supported by digital material and electronic communication ranging from the provision of slides or scripts in digital form to elaborate, interactive learning environments. In this respect sometimes pure e-learning scenarios and blended learning scenarios are distinguished. In [Henrich and Sieber, 2009] the authors argue that the more stable parts of the IR curriculum—perhaps the topics covered in courses IR *A* and IR *B* sketched in table 2—could be prepared for e-learning in a rich media format or in a text-based fashion. For the more unstable "special topics" the authors propose to combine digital presentation slides with some type of lecture recording, since these rapid e-learning techniques are more appropriate for content with a high rate of change.

As a further cornerstone of blended learning scenarios, applets are a good way of visualising important concepts and foster a deeper understanding by providing interactivity for students. On the other hand, applets without a clear didactic concept remain art for art's sake. In [Henrich and Sieber, 2009] three important aspects to be considered when designing an applet are mentioned: (1) The topic to be addressed has to be complex enough to justify the effort. If a figure can tell the story an applet might be overdone. (2) If there is no appealing idea for the visualisation, an applet is not the tool of choice. Furthermore, the visualisable aspects have to coincide with the aspects that should be clarified by the applet. (3) An applet should concentrate on a certain aspect or the relation between two aspects. If a significant amount of context is necessary, the focus of the applet may get lost.

As pointed out in section 3, lots of tools and applets exist and it would be an appealing idea to share these resources or at least to form a well maintained directory of IR related tools, applets, etc. A first step in this direction might be the *Teaching IR* subtree in the web site of FG-IR[35].

6 Conclusion

The importance of information retrieval has increased greatly in the last decades due to the fact that information and knowledge captured in documents is now a critical part of work and play for most people in the industrialised world. As a result, an increasing number of curricula in computer science and related fields now includes information retrieval as a subject. In this article, we have outlined which theoretical concepts we believe should be taught in IR, where we have presented different subsets for different groups of students. We have also addressed some of the practical questions that need to be answered when teaching IR at today's colleges and universities. Finally, we discussed different forms of teaching in the context of IR and when each form is appropriate.

Obviously, many points would be worth a controversial discussion, and we hope that this paper can help to stimulate this discussion.

[35] `http://www.fg-ir.de`, last visit: 19.1.09

References

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley.

[Bates, 1989] Bates, M. J. (1989). The design of browsing and berrypicking techniques for the online search interface. *Online Review*, 13(5):407–424. Online: `http://www.gseis.ucla.edu/faculty/bates/berrypicking.html`.

[Bates, 1990] Bates, M. J. (1990). Where should the person stop and the information search interface start? *Information Processing and Management*, 26(5):575–591.

[Bawde et al., 2007] Bawde, D., Bates, J., Steinerov, J., Vakkari, P., and Vilar, P. (2007). Information retrieval curricula: Contexts and perspectives. In *First International Workshop on Teaching and Learning of Information Retrieval (TLIR 2007)*, London, UK. The British Computer Society (BCS), Online: `http://www.bcs.org/upload/pdf/ewic_tl06_s3paper1.pdf`,.

[Belew, 2000] Belew, R. K. (2000). *Finding Out About: A cognitive perspective on search engine technology and the WWW*. Cambridge Univ. Press.

[Belkin, 1980] Belkin, N. J. (1980). Anomalous states of knowledge as a basis for information retrieval. *Canadian Journal of Information Science*, 5:133–143.

[Brin and Page, 1998] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117.

[Buckley and Lewit, 1985] Buckley, C. and Lewit, F. A. (1985). Optimization of inverted vector searches. In *8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 97–110, Montréal, Québec, Canada.

[Buckley and Voorhees, 2005] Buckley, C. and Voorhees, E. M. (2005). *TREC: Experiment and Evaluation in Information Retrieval*, chapter Retrieval System Evaluation, pages 53–75. Digital libraries and electronic publishing series. MIT Press, Cambridge, MA.

[Buschmann et al., 1996] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-oriented software architecture: a system of patterns*. John Wiley & Sons, Inc., New York, NY, USA.

[Cacheda et al., 2008] Cacheda, F., Fernandez, D., and Lopez, R. (2008). Experiences on a practical course of web information retrieval: Developing a search engine. In *Second International Workshop on Teaching and Learning of Information Retrieval (TLIR 2008)*, London, UK. The British Computer Society (BCS), Online: `http://www.bcs.org/upload/pdf/ewic_tl08_paper4.pdf`.

[Croft and Lafferty, 2003] Croft, B. and Lafferty, J., editors (2003). *Language Modeling for Information Retrieval*. Kluwer.

[Croft et al., 2009] Croft, B., Metzler, D., and Strohman, T. (2009). *Search Engines: Information Retrieval in Practice*. Pearson Higher Education, Old Tappan, NJ.

[Croft, 1995] Croft, W. B. (1995). What do people want from information retrieval? (the top 10 research issues for companies that use and sell IR systems). *D-Lib Magazine*, 1(5). `http://www.dlib.org/dlib/november95/11croft.html`.

[Dean and Ghemawat, 2004] Dean, J. and Ghemawat, S. (2004). Mapreduce: simplified data processing on large clusters. In *6th Symposium on Operating System Design & Implementation*, pages 137–150, Berkeley, CA, USA. USENIX.

[Deppisch, 1986] Deppisch, U. (1986). S-tree: a dynamic balanced signature index for office retrieval. In *9th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 77–87, Pisa, Italy.

[Ellis, 1989] Ellis, D. (1989). A behavioural approach to information retrieval system design. *Journal of Documentation*, 45(3):171–212.

[Ferber, 2003] Ferber, R. (2003). *Information Retrieval: Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web*. dpunkt, Heidelberg, 1. edition.

[Grossman and Frieder, 2004] Grossman, A. D. and Frieder, O. (2004). *Information Retrieval: Algorithms and Heuristics*, volume 15 of *The Information Retrieval Series*. Springer, Dordrecht, 2. edition.

[Hatcher and Gospodnetic, 2004] Hatcher, E. and Gospodnetic, O. (2004). *Lucene in Action (In Action series)*. Manning Publications Co., Greenwich, CT, USA.

[Hearst, 1999] Hearst, M. A. (1999). User interfaces and visualization. In [Baeza-Yates and Ribeiro-Neto, 1999].

[Henrich, 2008] Henrich, A. (2008). Information Retrieval 1: Grundlagen, Modelle und Anwendungen. University of Bamberg. Online: `http://www.uni-bamberg.de/minf/ir1_buch/`, last modified: 7.1.2008.

[Henrich and Sieber, 2009] Henrich, A. and Sieber, S. (2009). Blended learning and pure e-learning concepts for information retrieval: experiences and future directions. *Information Retrieval (Springer)*, 12.

[Ingwersen, 1992] Ingwersen, P. (1992). *Information Retrieval Interaction*. Taylor Graham, London.

[Jurafsky and Martin, 2008] Jurafsky, D. and Martin, J. (2008). *Speech and Language Processing*. Prentice Hall.

[Kuhlthau, 1988] Kuhlthau, C. C. (1988). Developing a model of the library search process: Cognitive and affective aspects. *Reference Quarterly*, 28(2):232–242.

[Lieberman, 1995] Lieberman, H. (1995). Letizia: An agent that assists Web browsing. In *International Joint Conference on Artificial Intelligence*, pages 924–929, Montréal, Québec, Canada.

[Liu and Croft, 2004] Liu, X. and Croft, W. B. (2004). Cluster-based retrieval using language models. In *27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 186–193, Sheffield, UK.

[Macdonald et al., 2008] Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., and White, W. R., editors (2008). *Advances in Information Retrieval: Proceedings of the 30th European Conference on IR Research (ECIR), Glasgow, UK*, volume 4956 of *LNCS*. Springer.

[Mandl, 2008] Mandl, T. (2008). Recent developments in the evaluation of information retrieval systems: Moving towards diversity and practical relevance. *Informatica*, 32:27–38.

[Mann, 2002] Mann, T. M. (2002). *Visualization of Search Results from the World Wide Web*. PhD thesis, University of Constance, `http://kops.ub.uni-konstanz.de/volltexte/2002/751/pdf/Dissertation_Thomas.M.Mann_2002.V.1.07.pdf`.

[Manning and Schütze, 2000] Manning, C. and Schütze, H. (2000). *Foundations of Statistical Natural Language Processing*. MIT Press.

[Manning et al., 2008] Manning, D. C., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge Univ. Press, Cambridge.

[Melucci and Hawking, 2006] Melucci, M. and Hawking, D. (2006). Introduction: A perspective on web information retrieval. *Information Retrieval (Springer)*, 9(2):119–122. `http://www.springerlink.com/content/32l13x402x276j14/`.

[Middleton and Baeza-Yates, 2007] Middleton, C. and Baeza-Yates, R. (2007). A comparison of open source search engines. Technical Report: `http://wrg.upf.edu/WRG/html/publications.html`.

[Moffat et al., 2005] Moffat, A., Zobel, J., and Hawking, D. (2005). Recommended reading for IR research students. *SIGIR Forum*, 39(2):3–14.

[Myaeng et al., 2008] Myaeng, S.-H., Oard, W. D., Sebastiani, F., Chua, T.-S., and Leong, M.-K., editors (2008). *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Singapore*.

[Riggs, 2002] Riggs, K. R. (2002). Exploring IR with Unix tools. *Journal of Computing Sciences in Colleges*, 17(4):179–194.

[Rijsbergen, 1979] Rijsbergen, C. J. v. (1979). *Information Retrieval*. Butterworth (Online: `http://www.dcs.gla.ac.uk/Keith/Preface.html`).

[Robertson, 2004] Robertson, S. (2004). Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation*, 60:503–520.

[Robertson, 2008] Robertson, S. (2008). On the history of evaluation in IR. *Journal of Information Science*, 34(4):439–456.

[Robertson and Sparck Jones, 1976] 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 et al., 1994] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M., and Gatford, M. (1994). Okapi at TREC-3. In *NIST Special Publication 500-226: Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–126.

[Rölleke et al., 2006] Rölleke, T., Tsikrika, T., and Kazai, G. (2006). A general matrix framework for modelling information retrieval. *Inf. Process. Management*, 42(1):4–30.

[Salton et al., 1975] Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

[Schaefer et al., 2005] Schaefer, A., Jordan, M., Klas, C.-P., and Fuhr, N. (2005). Active support for query formulation in virtual digital libraries: A case study with DAFFODIL. In Rauber, A., Christodoulakis, C., and Tjoa, A. M., editors, *Research and Advanced Technology for Digital Libraries. Proc. European Conference on Digital Libraries (ECDL 2005)*, LNCS. Springer.

[Shadbolt et al., 2006] Shadbolt, N., Berners-Lee, T., and Hall, W. (2006). The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101.

[Shneiderman, 1998] Shneiderman, B. (1998). *Designing the user interface*. Addison-Wesley.

[Shneiderman and Maes, 1997] Shneiderman, B. and Maes, P. (1997). Direct manipulation vs interface agents. *ACM interactions*, 4(6):42–61.

[Sparck Jones and Willett, 1997] Sparck Jones, K. and Willett, P. (1997). *Readings in information retrieval*. The Morgan Kaufmann series in multimedia information and systems. Morgan Kaufmann, San Francisco, CA.

[Stein, 2007] Stein, B. (2007). Principles of hash-based text retrieval. In *30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 527–534, Amsterdam, Netherlands.

[Stock, 2007] Stock, G. W. (2007). *Information Retrieval: Informationen suchen und finden*, volume 1 of *Einführung in die Informationswissenschaft*. Oldenbourg, München.

[Witten et al., 1999] Witten, I., Moffat, A., and Bell, T. (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Wong and Yao, 1995] Wong, S. K. M. and Yao, Y. (1995). On modeling information retrieval with probabilistic inference. *ACM Trans. Inf. Syst.*, 13(1):38–68.

Daniel Blank obtained the Dipl. degree in information systems in 2006 from the University of Bamberg. He is a phd student at the Chair of Media Informatics at the University of Bamberg. His research interests include peer-to-peer content-based image retrieval and geographic information retrieval.

Dipl.-Wirtsch.Inf. Daniel Blank
Otto-Friedrich-Universität Bamberg
Feldkirchenstraße 21, 96052 Bamberg
daniel.blank@uni-bamberg.de
www.uni-bamberg.de/minf/

Norbert Fuhr received a PhD (Dr.) in Computer Science from the Technical University of Darmstadt in 1986. He became Associate Professor in the computer science department of the University of Dortmund in 1991 and was appointed Full Professor for computer science at the University of Duisburg-Essen in 2002. His current research interests are retrieval models, digital libraries and interactive retrieval.

Prof. Dr. Norbert Fuhr
University of Duisburg-Essen, Campus Duisburg
Working group “Information Systems”
Department of Computational and Cognitive Sciences
Faculty of Engineering Sciences
47048 Duisburg
norbert.fuhr@uni-due.de
www.is.informatik.uni-duisburg.de/

Andreas Henrich obtained the Dipl. degree in information systems in 1987 from the University of Technology Darmstadt, the Dr. rer. nat. degree in 1990 from the University of Hagen, and the Venia Legendi in 1997 from the University of Siegen. Since 1998 he has been a Professor at the University of Bamberg and since 2004 he is a full Professor for Media Informatics at this university. His research interests are especially in information retrieval, data visualisation and exploration, and e-Learning.

Prof. Dr. Andreas Henrich
Otto-Friedrich-Universität Bamberg
Feldkirchenstraße 21, 96052 Bamberg
andreas.henrich@uni-bamberg.de
www.uni-bamberg.de/minf/

Thomas Mandl studied information and computer science at the University of Regensburg and at the University of Illinois. He worked as a research assistant at the Social Science Information Centre in Bonn and as assistant professor at the University of Hildesheim. He received a doctorate degree and a post doctoral degree (Habilitation) from the University of Hildesheim. His research interests include information retrieval, human-computer interaction and internationalisation of information technology.

PD Dr. Thomas Mandl
Information Science
University of Hildesheim
Marienburger Platz 22, 31141 Hildesheim
mandl@uni-hildesheim.de
www.uni-hildesheim.de/˜mandl

Thomas Rölleke is a senior lecturer and researcher at Queen Mary, University of London. He previously worked as product manager at Nixdorf Computer, lecturer at the University of Dortmund, and IT consultant. His research contributions include a probabilistic relational algebra, a probabilistic object-oriented logic, the relational Bayes, and foundations and theories of retrieval models. He pioneered HySpirit, a retrieval framework providing probabilistic reasoning and seamless DB+IR technology, and he is the founder of a start-up to market DB+IR.

Dr. Thomas Rölleke
Department of Computer Science
Queen Mary, University of London
London E1 4NS
thor@dcs.qmul.ac.uk
www.dcs.qmul.ac.uk/˜thor/

Hinrich Schütze received an MSCS in computer science from the University of Stuttgart in 1989 and a PhD in computational linguistics from Stanford University in 1995. After working at the Xerox Palo Alto Research Center and a number of Silicon Valley startups he returned to the University of Stuttgart as Chair of Theoretical Computational Linguistics in 2004. His research focuses on statistical natural language processing and information retrieval.

Prof. Dr. Hinrich Schütze
Universität Stuttgart, IMS
Azenbergstraße 12, 70174 Stuttgart
hs999@ifnlp.org
www.ims.uni-stuttgart.de/˜schuetze/

Benno Stein: Study at the Technical University of Karlsruhe. Dissertation and habilitation in computer science at the University of Paderborn. Appointed as full professor for Web Technology and Information Systems at the Bauhaus University Weimar in 2005. Research stays at IBM, Germany, and the International Computer Science Institute, Berkeley. His research focuses on the modelling and solving of knowledge-intensive information processing tasks.

Prof. Dr. Benno Maria Stein
Bauhaus-Universität Weimar
Faculty of Media / Media Systems
Bauhausstr. 11, 99423 Weimar
benno.stein@uni-weimar.de
www.webis.de