On Information Need and Categorizing Search

Faculty of Electrical Engineering, Computer Science, and Mathematics
Department of Computer Science
of the
University of Paderborn, Germany

The accepted dissertation of

Sven Meyer zu Eißen

in partial fulfillment of the requirements to obtain the academic degree of

Dr. rer. nat.

Reviewer:
Prof. Dr. Benno Stein
Prof. Dr. Norbert Fuhr
Prof. Dr. Hans Kleine Büning

Date of Oral Exam: February 23, 2007

Acknowledgements

I wish to thank Prof. Dr. Hans Kleine Büning for supporting me and my work. He provided a stimulating environment and chaired the reading committee. I want to express my gratitude to Prof. Dr. Benno Stein for supervising this thesis, and for numerous fruitful, serious, and enthusiastic discussions that initiated or improved many of the presented ideas and results. I am indepted to his guidance, patience, and mentorship.

Furthermore, I wish to thank Prof. Dr. Norbert Fuhr who evaluated this thesis and provided useful comments and suggestions for improvement.

For many valuable discussions and enjoyable breaks I want to thank all of my (former) colleagues, in particular Dr. Andreas Göbels, Oliver Kramer, Dr. Theo Lettmann, Dr. Oliver Niggemann, and Martin Potthast.

Finally I would like to thank all of my friends and my whole family, especially my wife Lisa.

Sven Meyer zu Eißen

Table of Contents

Notation

Symbol	Meaning
$\|\| \cdot \|\|$	norm
$\| \cdot \|$	cardinality or absolute value
$\langle \cdot, \cdot \rangle$	dot product
δ	Kronecker symbol, cluster distance, probability density
Δ	cluster diameter
φ	similarity function
λ, λ_i	capacity of minimum cut
Λ	weighted connectivity
θ	graph density
τ	labeling, labeling function
ρ	probability density function
$\overline{\rho}$	expected density
$\mathcal{C}$	clustering / categorization
$\mathcal{C}^*$	desired categorization / reference categorization
C, C_i	category or cluster, set of documents
d, d_i	document
$\mathbf{d}, \mathbf{d}_i$	vector or data structure for a document
$[\mathbf{d}]_j$	j-th component of vector $\mathbf{d}$
D	set of documents / document collection
D^*	desired set of documents
e	edge
E, E_i	edge set
$F, F_{i,j}$	F$-Measure$
$f_C(w)$	score for term w to be topic identifier in cluster C
$G = \langle V, E \rangle$	G is a graph with node set V and edge set E

$G(C_i)$	subgraph induced by the node set C_i
$H(C_j)$	entropy of C_j
$I(\mathcal{C})$	cluster validity index value for clustering $\mathcal{C}$
idf	inverse document frequency
$\mathbf{N}$	natural numbers
$prec$	precision
q	query
Q	set of queries
$\mathbf{R}$	real numbers
rec	recall
t_i	term
T, T'	set of terms
$\mathit{tf}(d, t_j)$	term frequency of term t_j within document d
$\mathit{tf}_D(t_j)$	term frequency of term t_j in collection D
$P(A \mid B)$	conditional probability for event A given B
$\mathit{tf} \cdot \mathit{idf}$	term frequency multiplied with inverse document frequency
V, V_i	node set
w, w_i	weight or edge weight

Preface

Information retrieval (IR) is a discipline that deals with the task of satisfying a person's information need with the help of a computer. IR systems let a user specify an information need, which is evaluated against large collections of digital documents.

Techniques for satisfying information needs have meanwhile become ubiquitous, be it in the form of search engines at home or at work, on mobile devices or on workstations, or as retrieval components in file systems, document repositories, databases, or knowledge management tools. The reason for this pervasiveness is the growing information need, the diversity of retrieval tasks, and the desired degree of personalization.

This thesis develops new concepts and new algorithms in various aspects throughout the information retrieval process. A focus lies on the automatic categorization of text and hypertext documents according to topic and genre. Aside from proposing new data structures and algorithms for these tasks, a novel IR process operationalization paradigm based on Model Driven Architecture is suggested.

All concepts and algorithms have been operationalized, among others within AIsearch, a search engine that has been awarded at the EASA-2004 (European Academic Software Award) competition.

Chapter 1

Introduction

The growing importance of information retrieval systems is accounted for not only by the Internet and digital libraries but also by the fact that vast numbers of companies organize their workflows and correspondence with digital documents. For bigger companies, the number of documents that have to be managed by an IR system easily goes into the millions. The cost of not finding a document within a company's intranet may be measured by the amount of money that has to be spent on recreating it. These costs can be significant; consider for example the cost to recreate a technical document that contains construction descriptions for parts of a technical system.

The key challenge in the development of IR systems is the adaption of search interfaces to human behaviour, limitations, and personal information needs. The following points illustrate the adaption challenge; each point asks questions about the design of an adequate retrieval process and also poses algorithmic problems.

- *Behavior.* Studies have shown that the majority of all keyword queries against Web search engines are one or two term[1] queries [127]. Aside from delivering a huge set of matching documents, short queries are often ambiguous since terms may have more than one meaning (polysemy). As a consequence, search results often comprise a large fraction of documents that do not fit the information needs of a user. On the other hand, relevant documents can contain terms that are semantically related but lexicographically distinct to a query term, and they will not be judged relevant if terms

[1]A term in this context denotes an atomic lexicographical unit, e.g. a word, a number, or an abbreviation.

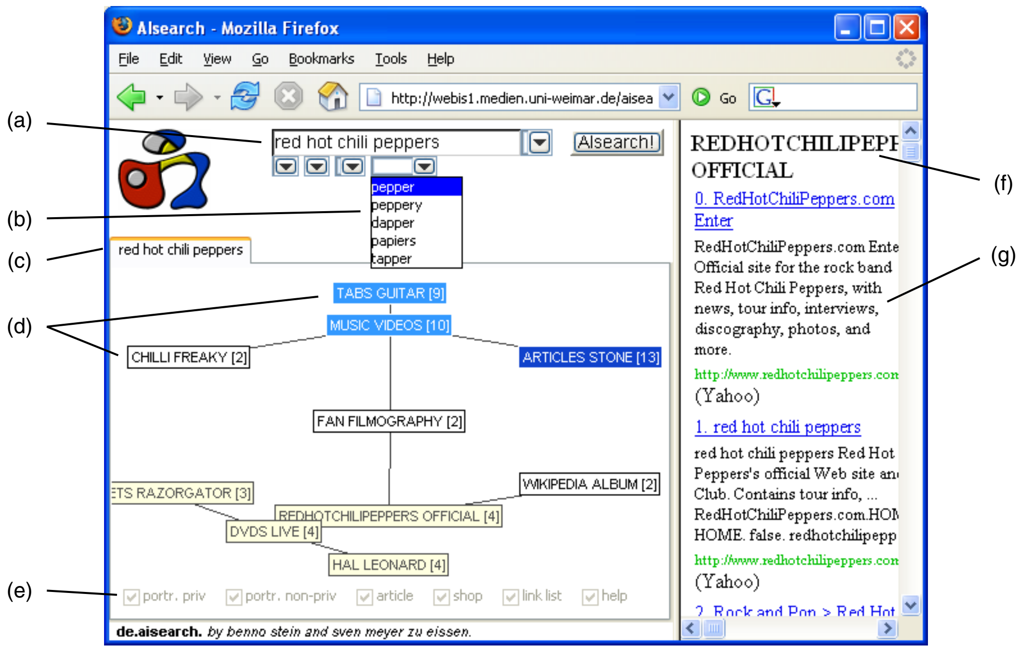
are matched exactly (synonymy). Another problem with short queries is that the query focus of a user is unclear; missing context information contributes to long result lists containing many irrelevant hits.

- *Limitations.* The number of matching documents for typical queries against very large document repositories goes easily into the thousands. Due to human limitations in information processing, only some of the found documents or extracted document snippets[2] can be read. Human information miners are hardly able to narrow down the search iteratively using different or more keywords since it is unclear in which context their query terms are used, and which terms could be useful to focus the search, for example to eliminate unwanted hits. Aside from being frustrating, a hand-crafted query reformulation is time consuming.

- *Information Need.* Search interfaces that follow the keyword search paradigm are broadly accepted by users; however, the challenge for the next decades is the development of search solutions that reflect a user's context. In other words, solutions that are able to (a) organize search results better than in the form of long lists, (b) filter beyond topic, i.e. to filter according to document quality and type, (c) adapt to a user's personal skills and experience concerning the underlying document collection, and (d) adapt to the retrieval task a user is concerned with—in short: to adapt to a user's information need.

This thesis contributes right here; techniques are developed which address the design and the operationalization of IR processes respecting the above points. The research focus is automatic document categorization, a technique which has been shown to improve retrieval performance.

[2]Document *snippets* are excerpts from documents that contain one or more query terms. In most of the well-known search engines, document snippets serve as preview for a found document.



REDHOTCHILIPEPPE OFFICIAL

0. RedHotChiliPeppers.com Enter

RedHotChiliPeppers.com Ente Official site for the rock band Red Hot Chili Peppers, with news, tour info, interviews, discography, photos, and more.

http://www.redhotchilipeppers.con (Yahoo)

1. red hot chili peppers

red hot chili peppers Red Hot Peppers's official Web site an Club. Contains tour info, ... RedHotChiliPeppers.com.HOM HOME. false. redhotchilipepp

http://www.redhotchilipeppers.con (Yahoo)

2. Rock and Pop > Red Hot

Figure 1.1: Screenshot of AIsearch. The area at the top (left) shows controls for query specification. Below, a generated category graph is shown in which each node represents a category. The edges indicate similarity relations: each category is connected with an edge to its most similar category. At the bottom, checkboxes for genre filtering are shown. On the right, a list of document snippets that is ordered by category is shown. The main text contains detailed explanations for the points (a) – (g).

AIsearch

The contributions of this thesis are manifest in our award-winning meta search engine AIsearch, which categorizes search results according to topic and genre. Figure 1.1 shows a screenshot of the AIsearch user interface: When query terms are specified in field (a), spelling corrections as well as alternative query terms are proposed in drop-down box (b). A categorization of the search results is shown in tab (c): Each node of the depicted graph represents a found category that is labeled with a topic identifier as well as with its size (d). Each category is connected by an edge to its most similar category. The checkboxes (e) allow to filter documents according to genre. On the right hand side, document snippets are shown (g), which are sorted by category (f). A mouse click on a category label selects the corresponding category: It centers the associated node within the display, and updates the result list with document snippets from the selected category.

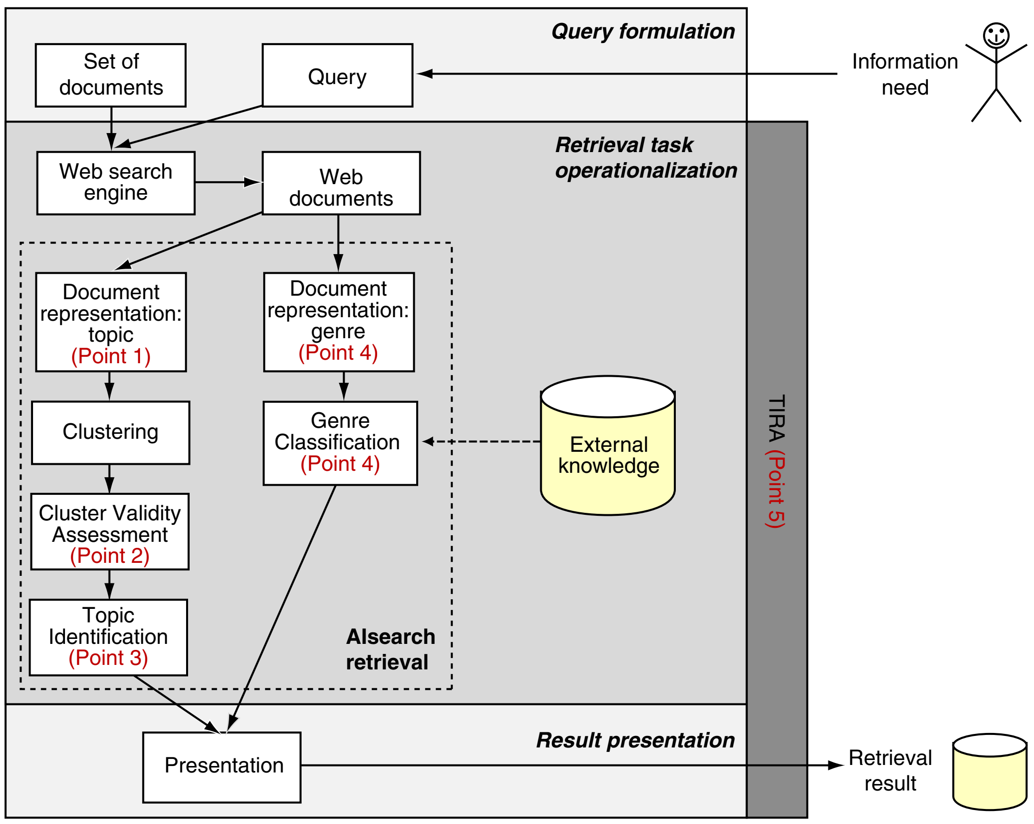

Query formulation
Set of documents
Query
Information need
Retrieval task operationalization
Web search engine
Web documents
Document representation: topic (Point 1)
Document representation: genre (Point 4)
Clustering
Genre Classification (Point 4)
External knowledge
Cluster Validity Assessment (Point 2)
Topic Identification (Point 3)
AIsearch retrieval
TIRA (Point 5)
Result presentation
Presentation
Retrieval result

Figure 1.2: The retrieval process underlying AIsearch. The numbers refer to the summarized contributions described in the text below.

1.1 Thesis Contributions

Figure 1.2 shows the building blocks of AIsearch: An information need is specified in the form of a query, which is processed by Web search engines. The delivered Web documents are converted to data structures for topic and genre categorization. The topic categorization step uncovers the underlying topic structure using cluster analysis and cluster validity tools. Each of the found categories is labeled with a topic identifier by our new WCC algorithm. A genre categorization allows to filter the resulting documents according to their form and function. The following points summarize the main contributions of this thesis; the numbering refers to the place within the AIsearch retrieval process in Figure 1.2.

(1) **Document Representation.** When regarding a document as a sequence of terms, the similarity between two documents with respect to topic can be quantified by the fraction of terms that both documents have in common. Traditional document representations like the Boolean model or the vector

space model are typical representatives that implement this view. However, these models fail to quantify to what extent the term order matches between documents. Since a term order match can reflect a concept match[3], the amount and the length of these matches constitute important similarity information. We propose the suffix tree document model along with new similarity measures that are able to quantify both aspects in parallel, term matches as well as term order matches. The computational complexity to implement the suffix tree document model is comparable to the Vector Space Model.

Information Need. The suffix tree document representation addresses a user's information need directly when browsing categories: Our experiments show that the suffix tree document representation substantially improves document categorization performance compared to traditional vector space representations. The performance gain is independent from the employed clustering algorithm.

(2) **Cluster Validity.** Past research on document categorization according to topic prevalently compares the maximum performance of clustering algorithms on given document collections. Although this research question is interesting, it is far away from realistic scenarios: The parameters of clustering algorithms are unknown when clustering search results on an ad hoc basis. In this connection we propose $\overline{\rho}$, a new cluster validity index for document clustering. The index $\overline{\rho}$ allows us to identify among a set of clusterings those which are generated using adequate parameters. An experimental evaluation shows that $\overline{\rho}$ delivers reliable results in comparison to existing approaches in document categorization scenarios.

Information Need. Since $\overline{\rho}$ allows to identify high-quality clusterings in an unsupervised manner, it is especially suited for post-retrieval categorization, allowing for user interfaces that display ad hoc categorizations of search results.

[3]Take for example compound nouns like "artificial intelligence" or text blocks containing domain specific speech such as typical phrases from financial reports or expressions from legal terminology.

(3) **Topic Identification.** When a categorization according to topic is determined using an unsupervised approach, it has to be presented to a user. In particular, the categories have to be labeled with characteristic terms for browsing. The question about which terms should be chosen for labeling is very difficult since a maximum of only five terms from the set of all terms within a document cluster can be presented to a user[4]. The labeling question is especially interesting since desired properties of category labels are partly contradictory. We introduce and formalize desired properties for category labels, and we propose the new WCC algorithm to compute them.

    Information Need. In contrast to the few existing approaches, which have until now been evaluated by human judgment, we develop an evaluation methodology that delivers reproducible results and allows for an absolute performance quantification in terms of precision and recall. Our experiments show that WCC identifies up to five terms per cluster with a very high precision when comparing them to human-assigned keywords. As an aside, a document corpus for experimenting has been developed, which is available upon request.

(4) **Genre Categorization.** Categorizations are grounded on meaningful criteria. In recent IR research, the term "categorization" is associated with an organization of documents according to *topic*. However, we will show that the *genre* of a document is a very useful categorization criterion when searching large document repositories. In particular, a first user study in the field of genre categorization has been conducted, which shows the great interest of human information miners to filter beyond topic.

    In particular, we propose a genre categorization scheme for Web documents, and we introduce a novel document representation that can be employed to classify documents according to genre. A feasibility study shows that genre categorization is possible, even in a large and heterogeneous collection like the World Wide Web.

    Information Need. To investigate the extent to which a genre categorization according to human wishes is feasible, we compiled a test corpus of Web

[4]Given that a typical cluster comprises about 2500 different term stems, there are $\binom{2500}{k}$ possibilities to choose a subset of k terms.

pages, which originally included 1209 Web pages and was recently extended to 1707 Web pages. The results for a multiclass classification with eight genre classes are very promising: the experiments show that about 80% of the corpus pages can be classified correctly. Meanwhile, the corpus has been used by other institutes for follow-up research.

(5) **Software Engineering.** The remaining contribution concerns software engineering. In particular, a Model Driven Architecture (MDA) approach for composing and executing IR processes is developed. In contrast to the commonly used library-based IR process design, our approach clearly separates the specification of an IR process from its operationalization. Our architecture called Tira allows to specify an IR process in the form of a platform-independent UML activity diagram, from which an executable platform-dependent model can be automatically generated.

Information Need. The proposed technology allows us to rapidly develop new prototypes, to adapt IR processes to personal preferences, and to test new ideas at the push of a button. In particular, Tira's possibility to tailor IR processes to fit individual skills, personal wishes, and retrieval scenarios contributes to satisfy a user's information need.

1.2 Thesis Overview

This chapter motivates the presented research and summarizes our contributions. Chapter 2 reviews the information retrieval process; in particular, data structures and algorithms for constructing document representations and for computing document similarity are presented.

Chapter 3 contains the main contributions related to categorization. After document categorization is motivated and existing research respecting the cluster hypothesis is summarized and compared, clustering algorithms for automatic ad-hoc category formation are reviewed. The following section surveys existing work in cluster validity and introduces our novel cluster validity index, $\overline{\rho}$. The next section introduces the suffix tree document model and presents theoretical as well as experimental comparisons to other approaches. The topic identification section introduces a formal framework of desired properties for cluster labels. After presenting the WCC algorithm for topic identification, a novel evaluation strategy is introduced and evaluation results are given. Finally, genre classification of Web pages is introduced as a novel instrument to satisfy a user's information need.

Our flexible software architecture for the composition and execution of IR processes, Tira, is introduced in Chapter 4. The thesis concludes with Chapter 5.

Chapter 2

Technical Background: Document Representation and Retrieval

Within the last years, the number of digital documents in the Internet, corporate intranets, and digital libraries exploded. IR systems evolved, which aim to provide structured access to large-scale document collections. From a user's perspective, these IR systems follow various search paradigms that broadly make up four categories [29]:

(1) *Unassisted keyword search.* Search terms are entered and the search engine returns a ranked list of document snippets. Representative: Google (`www.google.com`).

(2) *Assisted keyword search.* The search engine produces suggestions based on the user's initial query. Representative: AskJeeves (`www.askjeeves.com`).

(3) *Directory-based search.* Here, the information space is divided into a hierarchy of human-maintained categories, where the user navigates from broad to specific classes. Representative: Yahoo (`www.yahoo.com`).

(4) *Query-by-example.* The user selects an interesting document snippet, which is then used as the basis for a new query. Representative: AltaVista (`www.altavista.com`).

Search interfaces that implement these paradigms form the "visible" part of IR systems. In the following two chapters we will shed light on what happens in IR systems behind the scenes. The focus of this chapter is document representation, while the next chapter discusses document retrieval.

Table 2.1: Typical retrieval tasks.

	Retrieval task	Input	Task description
(1)	keyword search	keywords	Find documents that match the given keywords best.
(2)	local search	keywords, geographic location	Find documents that match the given keywords best and that relate to the given location.
(3)	similarity search	document	Find documents that are similar to the input document.
(4)	question answering	question in natural language	Find documents that answer the given question best. Preferably, generate a tailored answer.
(5)	plagiarism detection	document	Find passages within the document that are copied from other documents.
(6)	quality search	various	Find documents that comply with given quality criteria.

2.1 The Information Retrieval Process

The purpose of IR systems is to satisfy a user's information need. Typically, for a universe D of documents this task involves the identification of a subset $D^* \subset D$ of documents that are considered useful for satisfying a user's information need. Since information needs are manifold, IR systems are specialized correspondingly, i.e. they are designed to process specific retrieval tasks that arise from specific information needs (cf. Table 2.1, which resumes some examples).

Retrieval tasks are operationalized within a retrieval process that includes query formulation and result presentation (cf. Figure 2.1). The operationalization of retrieval tasks happens in the form of a retrieval function, which is defined as follows.

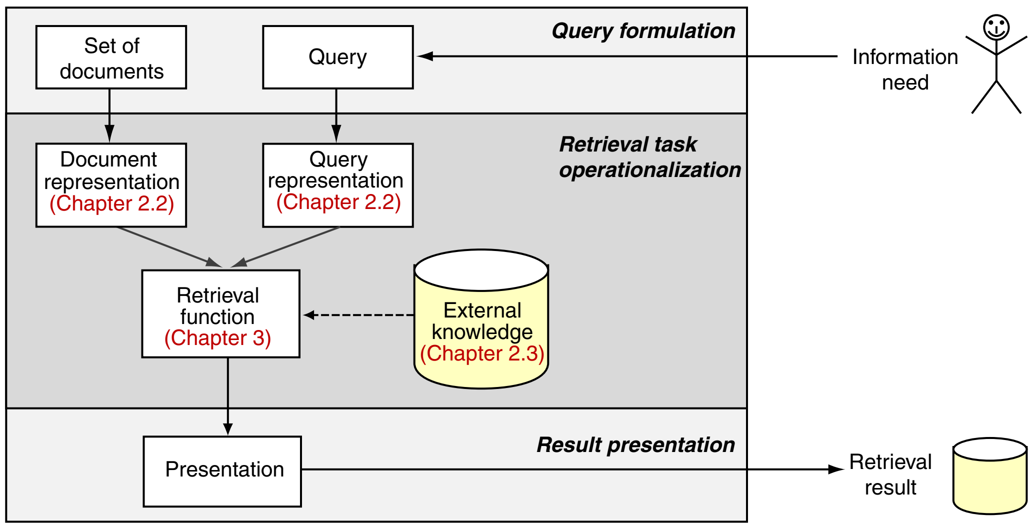

Query formulation
Set of documents
Query
Information need
Retrieval task operationalization
Document representation (Chapter 2.2)
Query representation (Chapter 2.2)
Retrieval function (Chapter 3)
External knowledge (Chapter 2.3)
Result presentation
Presentation
Retrieval result

Figure 2.1: The information retrieval process, adapted from [129]. A user formulates his/her information need in the form of a query (top), which is transformed into an internal representation, typically a tailored data structure. Likewise, documents from a collection are represented as data structures. A retrieval function operates on the internal representations to identify matching documents, probably using external knowledge (middle). Finally, the retrieval results are presented to the user (bottom).

Let $D = \{d_1, \ldots, d_n\}$ be a document collection, let q be a query from the set of queries Q, and let $\mathcal{K}$ be external knowledge[1]. A retrieval function $\rho_{D,\mathcal{K}} : Q \to \mathcal{R}$ is a mapping that selects a subset of D being relevant to a query $q \in Q$ given external knowledge $\mathcal{K}$. The result in $\mathcal{R}$ may take several forms, e. g. , $\mathcal{R}$ may be a ranked list, a categorization of the selected subset of D, or another representation that can be generated from D using $\mathcal{K}$. The operationalization of a retrieval function requires a representation of the documents d_i and the query q, which are discussed next.

2.2 Document Representation

A document representation $\mathbf{d}$ is a data structure to model a document within computer memory.[2] In literature, $\mathbf{d}$ is almost always a vector whose components

[1]$\mathcal{K}$ may contain thesauri, past queries, click stream statistics, etc. A discussion of the exploitable knowledge will follow at the end of this chapter.

[2]Bold latin lowercase letters, in general, denote vectors. We use the symbol $\mathbf{d}$ for a data structure here; the reason is that this data structure is almost always a vector in the IR literature.

quantify document properties like term frequencies. However, **d** may be another abstraction; in particular, we introduce the suffix tree document model in the next chapter.

Document representations are usually tailored to both, the retrieval task and the documents' information content. The following list grades document types according to information content.

(1) *Plain documents.* Documents that contain no more information than their text.

(2) *Structured documents.* Documents in which structure information like chapter boundaries or headlines is given. Example: Microsoft Word documents.

(3) *Hypertext documents.* Hypertext documents are structured documents that may contain references (links) to other documents. Example: HTML documents.

(4) *Annotated documents.* Documents that come along with meta-information like category or author. Examples are Semantic Web documents, which are enriched with ontological information [21, 20, 19].

The following rule of thumb applies: The more information a document contains, the merrier are the retrieval results, and the lower is the retrieval effort [129].

2.2.1 Term Vector Models

Let $D = \{d_1, \ldots, d_n\}$ be a document collection, and let $T = \{t_1, \ldots, t_m\}$ be the set of terms[3] each of which occurring in at least one $d \in D$. Moreover, let $\mathbf{R}$ denote the set of real numbers, and let the jth component of an n-dimensional vector $\mathbf{d} \in \mathbf{R}^n$ be denoted as $[\mathbf{d}]_j$.

[3] T is sometimes called the dictionary of D. The index numbers of the terms in T are arbitrary; i.e., no specific order of the t_i is assumed.

The Boolean Model. In the *Boolean document model*, the representation $\mathbf{d}_i$ of a document $d_i \in D$ is a vector whose j-th component indicates if t_j occurs in d_i, say, $\mathbf{d}_i$ is a boolean-valued vector with dimension m and $[\mathbf{d}_i]_j = 1$ iff $t_j \in d_i$. This kind of representation allows, among others, to quantify the similarity between two documents using a geometric measure, e.g. in the form of angles or distances between vectors in $\mathbf{R}^m$. An equivalent of the Boolean model is the set model, where a document is represented as a set in which the elements are the document's terms.

The Vector Space Model. A generalization of the Boolean model is the *Vector Space Model* (VSM) [121], in which the components of the vector representation are real values, the so-called term weights. Let $\mathit{tf}(d_i, t_j)$ denote the function that specifies how often the term t_j occurs in document d_i. Setting $[\mathbf{d}_i]_j = \mathit{tf}(d_i, t_j)$ includes term frequency information into the vector representation instead of boolean values only; the rationale is that terms that occur more often are more important [126].

However, with respect to a document collection one might argue that terms that are rather rare within the collection are important since they discriminate between documents, especially when the documents in the collection are on similar topics. Consider for example a document collection on feeding pets: two documents on cats are likely to be considered more similar than two documents on dogs and cats. Likewise, the term "feeding" loses its value within the collection since it will appear in most of the documents. In this connection *tf* is used in combination with the inverse document frequency *idf*. Let $\mathit{df}(t_j)$ denote the document frequency of a specific term t_j, say the number of documents in which t_j occurs. Then the inverse document frequency $\mathit{idf} : T \rightarrow \mathbf{R}_{+,0}$ with

$$\mathit{idf}(t_j) = \log\left(\frac{|D|}{\mathit{df}(t_j)}\right)$$

is a function that measures how a term t_j is distributed over the documents in D, in other words, it measures the discriminative power of t_j [126]. If only *idf* was used to weight terms in $\mathbf{d}_i$, then rare terms would dominate a geometric similarity computation: a term that occurs only once in the document collection has a maximum *idf* value. For this reason, $[\mathbf{d}_i]_j$ is chosen to be the product

$\mathit{tf}(d_i, t_j) \cdot \mathit{idf}(t_j)$. This product ensures that both very rare and very frequent terms do not influence a similarity computation too much[4].

The definition of *idf* can also be justified in the context of information theory. Let E_t be the probabilistic event that a term $t \in T$ occurs in a randomly drawn document $d \in D$. Then the probability of E_t is given as $P(E_t) = \frac{\mathit{df}(t)}{|D|}$. The amount of surprisal (also known as self-information [145, 154]) contained in this event depends only on the probability of the event. More specifically: the smaller this probability is, the larger is the surprisal associated with receiving information that the event actually occurred. A measure I that reflects surprisal is

$$I(E_t) = \log\left(\frac{1}{P(E_t)}\right) = \log\left(\frac{|D|}{\mathit{df}(t)}\right) = \mathit{idf}(t).$$

A property of I is that the surprisal of a composition of two mutually independent events equals the surprisal sum of the events, i.e. $I(E_{t_1} \cap E_{t_2}) = I(E_{t_1}) + I(E_{t_2})$. If the events E_t are regarded as atomic then they are independent by definition. However, two terms do not necessarily appear independent in a document since natural language comprises regular structures like figures of speech, compound nouns, and proper names that are made up of more than one term.

Consequently, when using I to measure surprisal, the underlying dictionary should reflect the independence condition: Instead of building the document representations based on single terms[5] from T, their construction based on a tailored *index term set* T' in which one element can comprise more than one term is advised. For example, in a collection on feeding pets it is a good idea to regard the compound noun "cat food" as one element of T' instead of two single terms. Section 2.2.3 discusses methods to construct T'.

VSM with Probabilistic Term Weights. A probabilistic method to define term weights is based on the assumption that so-called "specialty words", i.e. highly informative terms, are concentrated in a few documents within a collection [54, 55, 6]. Given this assumption, the value of a term can be quantified using the "divergence from randomness" theory, which generalizes Ponte and Croft's language modeling approach [104] as follows.

[4]More recent term weighting approaches factor document lengths into term weights. An approach that has shown superior performance for queries in TREC collecions is the BM25 term weighting scheme [113, 114].

[5]Single terms are also referred to as unigrams.

Let $D = \{d_1, \ldots, d_n\}$ be a document collection, let t denote a term and let $tf_D(t)$ denote the number of occurrences of t in D. For each of the $tf_D(t)$ copies of t, a random process can be imagined that first selects a $d \in D$ at random and then assigns t to d. Since "nonspecialty words" are assumed to be distributed uniformly at random over the n documents, the probability for k occurrences from the $tf_D(t)$ copies of t in a single document can be modeled using a Bernoulli experiment:

$$P(X_t = k) = B(n, tf_D(t), k) = \binom{tf_D(t)}{k} p^k q^{tf_D(t)-k} \tag{2.1}$$

where $p = 1/n$ and $q = (n-1)/n$. Here, the random variable X_t describes the term frequency of t in a document from D. Again, this model assumes term independence.

The application of this model is as follows. Using the formula above[6], we can compute for a given document $d_i \in D$ the probabilities $P(X_t = k)$ for each term, using $k = tf(d_i, t)$. The lower this probability, the less the term t is distributed in accordance with the model, and the more informative it is. Consequently, a function that increases monotonously as P decreases quantifies the informative value of a term. Since term probabilities also depend on a document's length, Amati and van Rijsbergen propose to "normalize" k according to document length first. In particular they propose to choose

$$\bar{k}(d_i, t) := tf(d_i, t) \cdot \log_2\left(1 + \frac{avg(D)}{l(d_i)}\right)$$

Here, $l(d_i)$ denotes the length of document d_i measured by the number of terms that d_i contains, and $avg(D) = \frac{1}{n}\sum_{d_i \in D} l(d_i)$ denoting the average document length within D.

A weight function that increases monotonously as P decreases and, consequently, grows with a term's importance in d_i is again the surprisal, resulting in

$$w_1(d_i, t) := -\log_2(P(X_t = \bar{k}(d_i, t)))$$

Amati and van Rijsbergen propose to choose as term weight a product $w(d_i, t) =$

[6]In practice, the mentioned probabilities are approximated using a Poisson distribution.

$w_1(d_i, t) \cdot w_2(d_i, t)$, where w_2 is a smoothing function that quantifies the information gain of a term within the so-called "elite set" $S_t \subseteq D$, say, the set of documents in which t occurs. The authors propose two different weighting schemes for $w_2(t)$, one based on Laplace's law of succession ($w_2(d_i, t) := \frac{1}{k(d_i,t)+1}$), and another as the ratio of two Bernoulli processes ($w_2(d_i, t) := \frac{tf_{S_t}(t)+1}{|S_t| \cdot (k(d_i,t)+1)}$).

It should be noted that the model of randomness, which is implemented in Formula (2.1) is interchangeable: Instead of using a Binomial distribution, models based on Bose-Einstein-statistics, the geometric distribution, and $tf \cdot idf$ can be employed [6, 5].

2.2.2 Document Similarity

Document representations are usually designed such that a function φ can be stated that maps from the representations $\mathbf{d}_1$ and $\mathbf{d}_2$ of two documents d_1, d_2 onto the interval $[0; 1]$ and that has the following property: If $\varphi(\mathbf{d}_1, \mathbf{d}_2)$ is close to one then the documents d_1 and d_2 are similar; likewise, a value close to zero indicates a high dissimilarity. Formally, a function φ is called similarity function if it fulfills at least point (1) of the following properties [152]. However, if not stated otherwise, we assume a similarity function to satisfy points (1)-(4).

(1) $\forall d_i \in D$: $\varphi(\mathbf{d}_i, \mathbf{d}_i) > 0$

(2) $\forall d_i, d_j \in D$: $\varphi(\mathbf{d}_i, \mathbf{d}_j) \geq 0$ (non-negativity)

(3) $\forall d_i, d_j \in D$: $\varphi(\mathbf{d}_i, \mathbf{d}_j) = \varphi(\mathbf{d}_j, \mathbf{d}_i)$ (symmetry)

(4) $\forall d_i, d_j \in D$: $\varphi(\mathbf{d}_i, \mathbf{d}_j) \in [0, 1]$ (normalization)

(5) $\forall d_i, d_j, d_k \in D$: $\varphi(\mathbf{d}_i, \mathbf{d}_j) + \varphi(\mathbf{d}_j, \mathbf{d}_k) \geq \varphi(\mathbf{d}_i, \mathbf{d}_k)$ (triangle inequality)

The applications of a similarity function φ are manifold. A query q in the form of a keyword list can be regarded as a very small document; consequently, φ can serve to define a retrieval function for similarity search. Moreover, φ can be used to generate similarity graphs, which in turn are a basis for clustering. The function φ can also be turned into a dissimilarity function by computing $1 - \varphi$ if φ is normalized and $\varphi(\mathbf{d}_i, \mathbf{d}_i) = 1$ for all $\mathbf{d}_i$. In particular, dissimilarity matrices enable multidimensional scaling to visualize document collections [131, 18, 77].

Since φ operates on the abstractions of the documents d_i it must be tailored to the representations $\mathbf{d}_i$. The remainder of this subsection gives an overview of similarity measures for set-based, geometric, and probabilistic document representations.

Set-based Similarity Measures. Given that documents $d_i, d_j \in D$ are represented as sets D_i, D_j of their terms, similarity can be quantified by set intersection ratios. In particular, the Jaccard coefficient φ_{jacc}, the cosine coefficient $\varphi_{coscoef}$, the dice coefficient φ_{dice} and the overlap coefficient φ_{over} have been employed in the past to measure set similarity, and they are defined as follows [111].

$$\begin{aligned}
\varphi_{jacc}(D_i, D_j) &= \frac{\mid D_i \cap D_j \mid}{\mid D_i \cup D_j \mid} \\
\varphi_{coscoef}(D_i, D_j) &= \frac{\mid D_i \cap D_j \mid}{\mid D_i \mid^{1/2} \cdot \mid D_j \mid^{1/2}} \\
\varphi_{dice}(D_i, D_j) &= \frac{\mid D_i \cap D_j \mid}{\mid D_i \mid + \mid D_j \mid} \\
\varphi_{over}(D_i, D_j) &= \frac{\mid D_i \cap D_j \mid}{\max(\mid D_i \mid, \mid D_j \mid)}
\end{aligned}$$

It should be noted that generalizations of these coefficients exist, which are designed to incorporate term weights. Put another way, these generalizations apply for geometric similarity measurement.

Geometric Similarity Measures. The most popular similarity measure for term vector representations is the cosine similarity function. Let $\mathbf{d}_i, \mathbf{d}_j$ denote the vector representations of the documents d_i and d_j respectively. The cosine similarity function $\varphi_{cos} : \mathbf{R}^m \times \mathbf{R}^m \rightarrow [0, 1]$, which measures the cosine of the angle between $\mathbf{d}_i$ and $\mathbf{d}_j$ is defined as

$$\varphi_{cos}(\mathbf{d}_i, \mathbf{d}_j) = \frac{\langle \mathbf{d}_i, \mathbf{d}_j \rangle}{\|\mathbf{d}_i\| \cdot \|\mathbf{d}_j\|}$$

where $\langle \cdot, \cdot \rangle : \mathbf{R}^n \times \mathbf{R}^n \rightarrow \mathbf{R}$ denotes the dot product and $\| \cdot \| : \mathbf{R}^n \rightarrow \mathbf{R}$ denotes the Euclidean norm. In comparison to a linear relationship between angle and similarity, the cosine similarity values are amplified, especially for angles in

$[0, \pi/4]$ (cf. Figure 2.2). This effect is considered useful when sparse vectors are given and the similarity values tend to be low. Another property of the cosine similarity measure is its scale invariance, i.e. $\varphi(\alpha \cdot \mathbf{d}_i, \beta \cdot \mathbf{d}_j) = \varphi(\mathbf{d}_i, \mathbf{d}_j)$ for all scalars α, β. The interpretation of scale invariance in the document domain is that documents are regarded as equal when they use the same terms in the same proportion; document length is not important.

Another natural similarity measure derives from the Euclidean distance between $\mathbf{d}_i$ and $\mathbf{d}_j$. With $d_{max} = \max_{i,j} \|\mathbf{d}_i - \mathbf{d}_j\|$ a similarity measure can be defined as $1 - \frac{\|\mathbf{d}_i - \mathbf{d}_j\|}{d_{max}}$ or, without the need to compute d_{max}, as $\frac{1}{1+\|\mathbf{d}_i - \mathbf{d}_j\|}$. However, these measures are not scale invariant.

A scale invariant Euclidean distance measure operates in the unit sphere, when the term vectors are scaled to length 1. Let $\mathbf{d}'_i$ denote these scaled vectors, say, $\mathbf{d}'_i = \frac{\mathbf{d}_i}{\|\mathbf{d}_i\|}$. Since the maximum distance between two points in the first quadrant of the unit sphere in $\mathbf{R}^n$ is $\sqrt{2}$, a similarity measure based on the Euclidean distance is

$$\varphi_{eucl}(\mathbf{d}_i, \mathbf{d}_j) = 1 - \frac{1}{\sqrt{2}} \cdot \|\mathbf{d}'_i - \mathbf{d}'_j\|$$

Note that φ_{eucl} directly relates to φ_{cos} since distances between points on the unit sphere are determined by the angle between the points. In particular,

$$\begin{aligned} \|\mathbf{d}'_i - \mathbf{d}'_j\| &= \sqrt{\sin^2(\alpha) + (1 - \cos(\alpha))^2} \\ &= \sqrt{\sin^2(\alpha) + 1 - 2\cos(\alpha) + \cos^2(\alpha)} \\ &= \sqrt{2 - 2\cos(\alpha)} \end{aligned}$$

where α denotes the angle between $\mathbf{d}'_i$ and $\mathbf{d}'_j$. Consequently, $\varphi_{eucl}(\mathbf{d}_i, \mathbf{d}_j) = 1 - \sqrt{1 - \cos(\alpha)}$.

Strehl and Ghosh propose the similarity measure $\varphi_{exp} = e^{-\|\mathbf{d}_i - \mathbf{d}_j\|^2}$, which is also based on Euclidean distance and has useful properties when clustering with k-Means [141]. Another measure that has not been used yet is $1 - \sin(\alpha)$, which is scale invariant and has the salient property that it reduces noise since it scales down low similarity values.

Figure 2.2 illustrates similarity function values for normalized vectors in the first quadrant of $\mathbf{R}^n$ depending on the angle between them. Like the curves show,

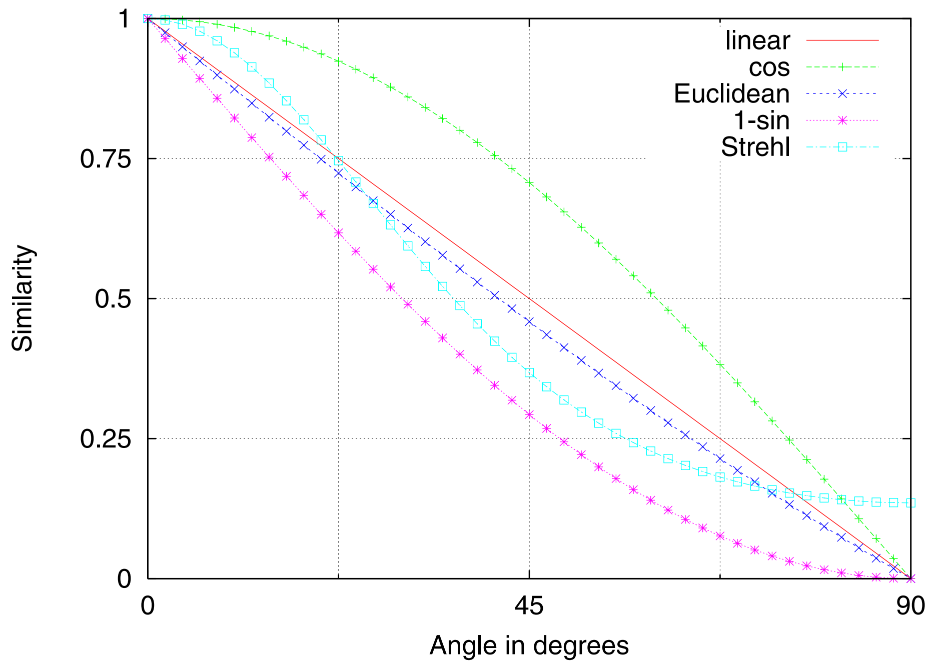


Figure 2.2: Plots of geometric similarity function values (y-axis) for normalized vectors in the first quadrant of $\mathbf{R}^n$ depending on the angle between them (x-axis).

each of them has different characteristics concerning the dilution and amplification of similarity values.

Probabilistic Similarity Measures. Let $P(R \mid d_i)$ denote the probability that a user finds $d_i \in D$ relevant to a query q. Probabilistic retrieval models employ estimations of $P(R \mid d_i)$ to sort the documents in D with respect to relevance to q [46][7]. The estimations are based on Bayes' theorem, namely

$$P(R \mid d_i) = \frac{P(d_i \mid R) \cdot P(R)}{P(d_i)}$$

Here, $P(d_i \mid R)$ denotes the probability for selecting d_i from the set of relevant documents; likewise, $P(d_i)$ is the probability to randomly draw d_i from the set of all documents. Since the latter probability is constant for a collection, the

[7]In Fuhr and Buckley's original work [47], odds in the form $O(R, d_i) = \frac{P(R|d_i)}{P(\overline{R}|d_i)}$ were used for ranking instead of probabilities for similarity computation. If $O(R, d_i) > 1$ then the probability for d_i being relevant is greater than d_i not being relevant. This rule, also known as Bayesian decision rule, enables an IR system to decide how many results should be retrieved. Moreover, the Bayesian decision rule minimizes the expected error.

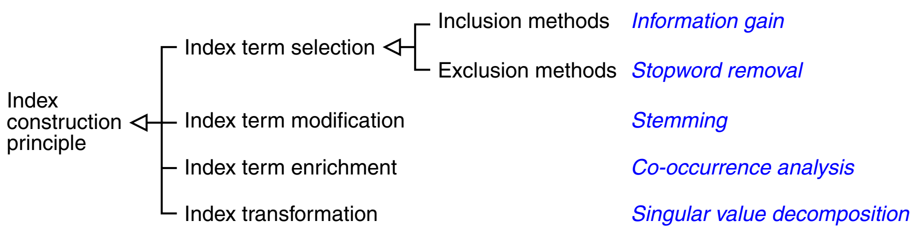


Figure 2.3: A taxonomy of index construction principles for vector space models.

former probability $P(d_i \mid R)$ remains to be estimated. These estimations are based on the Boolean vector representation and the assumption that terms occur independently in documents. The estimated probabilities can be adapted to a user's information need, which must be given in the form of relevance feedback for retrieved documents. Details can be found in [47, 46].

2.2.3 Index Term Set Construction Methods

Up to now T was defined to be the set of all terms that occur in at least one $d \in D$. However, the composition of document models from a tailored *index term set* T' can significantly enhance the performance of tasks like similarity computation or classification. From a linguistic point of view two documents are topically equal when they share the same concepts rather than the same terms. From a probabilistic point of view, the index terms in T' should appear independently (cf. Section 2.2.1). From an algebraic point of view this means that the index terms in T' should be used to create an orthogonal basis from which the document vectors are composed as linear combinations.

Generally, the index term set construction problem is a specialization of the well-known feature selection problem in machine learning [87, 66, 76, 15]. Since the evaluation of each possible subset of T with respect to the outlined goals is computationally infeasible, the construction of T' follows heuristics that exploit specific text domain knowledge. In the following, methods to construct T' by selection, modification, enrichment, and transformation of terms are resumed; each of them strives to reflect the mentioned goals. Figure 2.3 depicts a taxonomy of index term set construction methods.

Index Term Selection. The methods that operationalize index term selection subdivide into exclusion and inclusion methods. The goal of exclusion methods is to remove terms from T that simply add noise when computing a similarity value. Examples for these terms include common words like articles, prepositions or conjunctions (e.g. “the”, “and”, “or”, “of”, “from” etc.) that certainly do not contain topical information. The *tf* · *idf* term weighting scheme would adjusts these terms’ weights to a value close to zero automatically; however, these so-called stopwords can already be identified and removed accurately at parsing time according to stopword lists. This procedure speeds up similarity computation and saves memory in the document representations.

In contrast to exclusion methods, which construct T' in a bottom-up manner by removing terms from T, inclusion methods construct T' from scratch, say, in a top-down manner. As outlined above, good index terms reflect concepts and may be composed out of several terms. If the occurrence of terms correlates, e.g. within a fixed-size sliding window, the terms may be considered as concept and added to T' [89]. Another way to identify concepts in the form of related terms is to measure their mutual information [147] or to use association rule mining [3, 4]. However, these methods’ time complexity is in general at least $O(|T|^2)$. The suffix tree document model [96] contributes right here: it allows for non-trivial index term set construction in linear time.

Index Term Modification. Term modification is necessary to map morphologically different words that embody the same concept to the same index term. E.g., the terms `connection`, `connecting`, `connectivity`, `connector` are variants of `connect`, and they should map to the same index term. Stemming algorithms apply here; their goal is to find canonical forms for inflected or derived words, e.g. declined nouns or conjugated verbs. Since the adaption of words to gender, number, time, and case are language-specific, it might be reasonable to compile a set of rules that map an inflected word back to its stem. This is the way how Porter’s stemmer for the English language works [106]. However, rule-based stemming algorithms require the development of specialized rule sets for every language.

Statistical stemming algorithms generate stemming dictionaries based on large document collections. For this purpose, all terms from a document collection are inserted character-wise into a tree whose edges are labeled with the words’ characters. When a term is inserted, the path emanating from the root is followed while its current edge label matches the word's current character. In case of a mismatch, a new edge is inserted and labeled with the actual character.

If the number of a node's successors at some minimum depth is large enough then the character concatenation on the path from the root to this node is a candidate stem. Several techniques for identifying candidate stems have been proposed in the past; variants are based on thresholds, relative successor variety, and entropy [51, 43, 44]. Besides their language independence, statistical stemmers have the ability to adapt to document-specific vocabulary or language changes. Note that these stemmers are not only able to identify stems that originate from suffix removal, but that they can also be used to remove prefixes. For example, if the suffix `connect` from the term `reconnected` is inserted into the suffix tree, a match along the path that is associated with `connect` will deliver a high confidence hint for `connect` being a correct stem. A recent study on statistical stemming can be found in [140].

Index Term Enrichment. We classify a method as term enriching if it introduces terms not found in T. By nature, meaningful index term enrichment must be semantically motivated and exploit linguistic knowledge. A standard approach is the possibly transitive extension of T by synonyms, hyponyms, and co-occurring terms [59, 137]. The extension shall alleviate the problem of different writing styles, or of vocabulary variations observed in very small document snippets as they are returned from search engines.

Note that these methods are not employed to address the problem of polysemy, since the required in-depth analysis of the term context is computationally too expensive for many similarity search applications.

Index Transformation. Let $D = \{d_1, \ldots, d_n\}$ be a document collection over the term set $T = \{t_1, \ldots, t_m\}$. An $m \times n$ matrix A whose columns consist of the document models $\mathbf{d}_i$ is called *term-document matrix* for D. We understand index transformation as a function f that projects A to a $k \times n$ matrix A' with the goal to enhance the document models with respect to a retrieval task. In contrast to the above mentioned index construction methods, which operate on single columns of the term-document matrix, index transformation methods operate on the entire matrix.

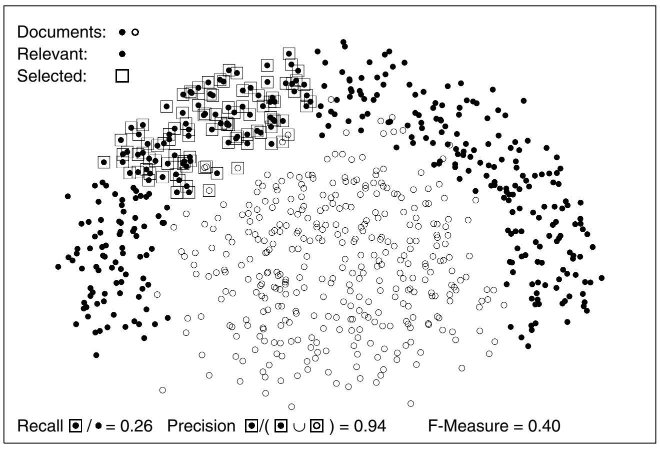


Figure 2.4: Illustration of precision and recall. The figure shows documents of two true classes: The solid points symbolize documents that are relevant to a query, and the outlined points symbolize irrelevant documents. The boxed points stand for documents that a retrieval system has selected for presentation. The set of boxed points shows a selection of high precision, which consists to 94% of relevant documents. Its recall is only about 0.26 since only 26% of all relevant documents are selected.

A popular representative index transformation method is latent semantic indexing (LSI) [27, 11, 101], which uses a singular value decomposition of the term-document matrix in order to improve query rankings and similarity computations. For this purpose, the document vectors as well as the queries are projected into a low-dimensional space that is spanned by the eigenvectors of $A^T A$ that correspond to the k largest eigenvalues $\sigma_1, \dots, \sigma_k$.

2.3 Techniques for Information Need Satisfaction

An IR system is considered successful when a user can formulate his or her information need easily and when the retrieval results point a user directly to relevant documents. In our context, where a retrieval system selects a subset $D' \subset D$ as being relevant, the performance is often measured in terms of precision and recall with respect to the "true" set of relevant documents, D^* [111]: Precision denotes the fraction of relevant documents within a retrieval result D', while recall denotes the fraction of retrieved relevant documents with respect to all relevant documents within a collection (cf. Figure 2.4). I. e.,

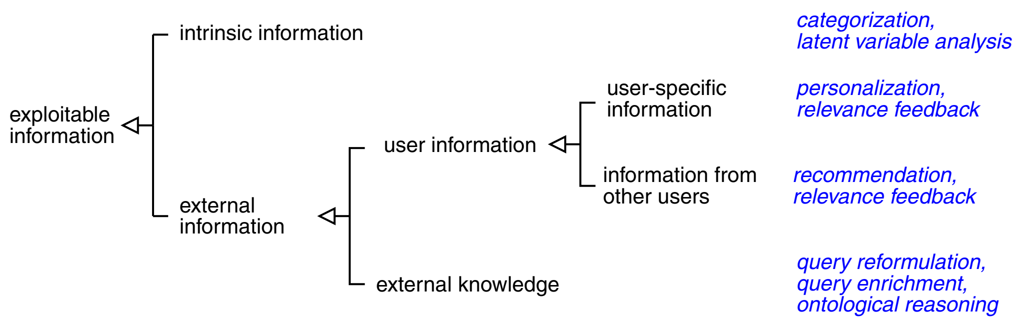

exploitable information
intrinsic information
external information
user information
external knowledge
user-specific information
information from other users
categorization, latent variable analysis
personalization, relevance feedback
recommendation, relevance feedback
query reformulation, query enrichment, ontological reasoning

Figure 2.5: Taxonomy of information resources available to a retrieval system (left) with example technologies that operationalize information exploitation (right).

$$prec = \frac{\mid D' \cap D^* \mid}{\mid D' \mid} \text{ and } rec = \frac{\mid D' \cap D^* \mid}{\mid D^* \mid}$$

An IR system's main task is to maximize both, precision and recall of retrieval results[8]. Several information sources can be exploited to allow for good performance: (a) Intrinsic information (the document collection itself) (b) external information from one or more users (recorded queries or click streams etc.), or external knowledge like ontologies or thesauri. Figure 2.5 shows an information taxonomy along with example technologies that operationalize information exploitation. In the following, some technologies are sketched to exemplify how information is exploited.

- *Personalization.* A personalized IR system builds a user profile based on an application-specific user model throughout one or more search sessions. These profiles are employed to rank and filter the results for upcoming queries for the same user and associated context.

- *Recommendation.* Recommender systems collect information like relevance judgments from a multitude of users in a specific context and use this information to rank, filter, or augment query results [2].

- *Relevance Feedback.* Systems like Rocchio [120] let a user judge which of the previously retrieved documents are relevant or irrelevant for her or his information need. These judgments may be used to automatically gener-

[8]The simultaneous maximization of precision and recall is an ambitious goal: Recall can be enlarged when delivering more documents. However, at the same time precision tends to get smaller unless a perfect retrieval function is given.

ate enhanced queries or to better rank and filter retrieval results in future queries.

- *Categorization.* Clustering algorithms aim to uncover the intrinsic category scheme of a document collection, enabling a user to focus the search on selected categories.

These technologies are useful to a greater or lesser extent depending on the individual use case, which may be characterized by aspects like collection size, user skill, and background knowledge. In this connection a user's skill denotes his or her ability to formulate or reformulate a query q; a user's background knowledge relates to the collection D and the domain with which his or her query is associated[9]. Table 2.2 summarizes the impact of the outlined approaches with respect to the mentioned aspects.

	huge collection	small collection	high formul. skill	low formul. skill	good knowledge of D	bad knowledge of D
Personalization	++	o	+	++	o	++
Recommendation	+	+	+	+	o	++
Relevance Feedback	+	++	-	++	o	+
Categorization	++	+	-	++	o	++

Table 2.2: Technology impact vs. collection and user characteristics. The performance impact estimations are tendencies oriented at current publications in the field.

[9]These aspects are subsumed by the concept *bounded rationality*, which "is used to designate rational choice that takes into account the cognitive limitations of both, knowledge and cognitive capacity" [156].

Chapter 3

Information Need and Categorizing Search

Search engine users know the surprising experience to learn from result lists about the diversity of topics that are associated with particular query terms. A reason for this surprisal is that a user's mental model on topics and related query terms does not match the real-world model: either, the query terms do not reflect a user's information need since they are too general, too narrow, or simply inadequate, or some of the contexts in which the query terms are used are unknown in advance. In each case, a summarizing overview of identified document categories would be helpful to navigate within search results and to figure out how the original query can be refined.

Uncovering category structures of a document collection is the main contribution of this chapter. First, two categorization paradigms are discussed, and a rationale for document clustering is given [133]. We then review algorithms for automatic category formation and for the validation of clusterings. In particular, we introduce the expected density $\overline{\rho}$ as a robust tool to assess document cluster quality [138] and argue why our experiments are the first that reflect a realistic scenario in unsupervised document categorization [90]. Moreover, we contribute the suffix tree document model, which will prove to better reflect document similarity than traditional approaches, and consequently to be an excellent foundation for cluster analysis [96].

Once a meaningful clustering is found, it has to be presented to a user. In this connection, each cluster has to be labeled with characteristic terms that

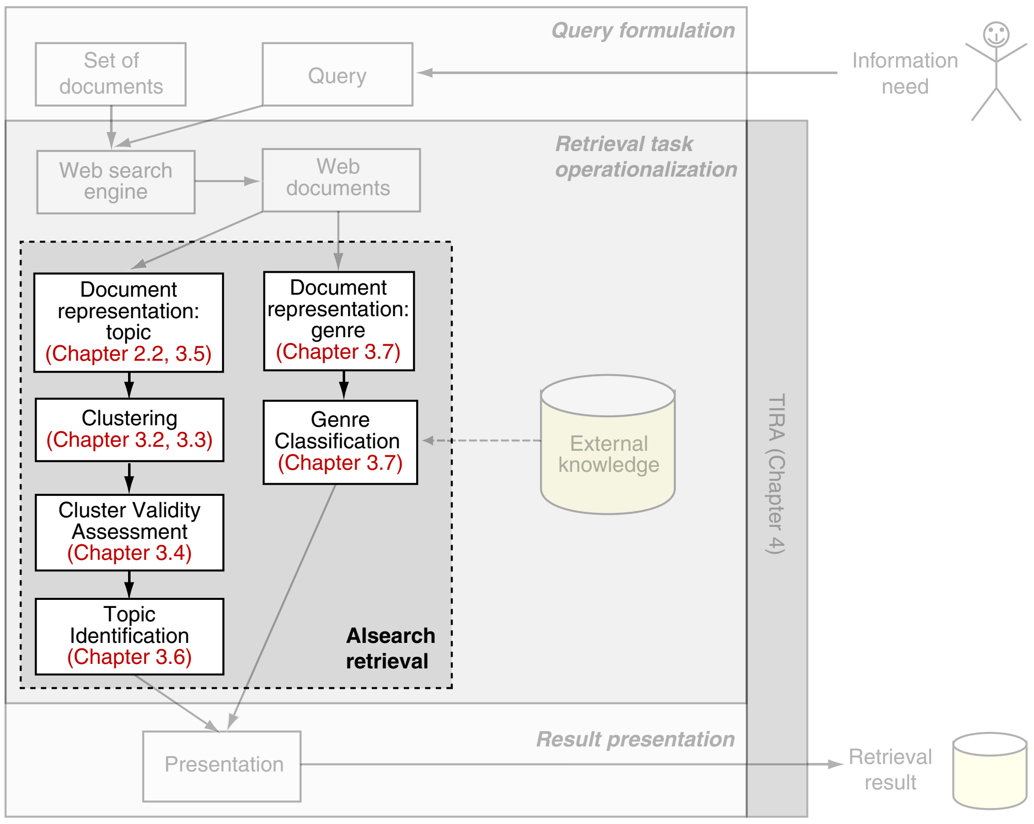

Query formulation
Set of documents
Query
Information need
Retrieval task operationalization
Web search engine
Web documents
Document representation: topic (Chapter 2.2, 3.5)
Document representation: genre (Chapter 3.7)
Clustering (Chapter 3.2, 3.3)
Genre Classification (Chapter 3.7)
External knowledge
Cluster Validity Assessment (Chapter 3.4)
Topic Identification (Chapter 3.6)
Alsearch retrieval
TIRA (Chapter 4)
Result presentation
Presentation
Retrieval result

Figure 3.1: Chapter organization.

characterize the cluster's contents. Finding good cluster labels is a challenging task since the labels should have properties like being summarizing and discriminating. These and other cluster label properties are discussed, and an algorithm for cluster labeling called WCC is given [138]. The experiments, which are the first ones that are reproducible in this area, show that WCC outperforms other approaches.

Finally, we propose a genre category scheme for Web documents. In contrast to topic categories, which reflect a document's thematic context, genre categories relate to a document's form and presentation; typical Web genres include online shop, private home page, link list, discussion forum, and article. Web pages of a specific genre can up to now not be specified to be included or excluded from search results—however, this aspect constitutes an important part of a user's information need. In particular, we discuss different kinds of genres that can be found in the Web, we present a user study on genre usefulness, and we propose a document model for genre analysis along with a categorization algorithm [92].

The aforementioned techniques were put to work in the form of our meta search engine AIsearch: We used indices of popular search engines as a starting point; a subsequent processing shall improve the retrieval quality employing the aforementioned tools [91, 93]. Figure 3.1 depicts this chapter's structure, which is oriented at the AIsearch retrieval task organization described at the outset.

3.1 Supervised vs. Unsupervised Categorization

A categorization $\mathcal{C} = \{C_1, \ldots, C_k\}$ is a partition of a document collection D into subsets $C_1, \ldots, C_k \subseteq D$ such that $\cup_{i=1}^{k} C_k = D$. A categorization is called *exclusive* if a document falls into exactly one of the categories, that is, $C_i \cap C_j = \emptyset$ for $i \neq j$.

Automatic categorization aims to discover the intrinsic category structure $\mathcal{C}^*$ of D, which is optimal with respect to criteria like topics[1] from the human perspective. The employed algorithms are either supervised or unsupervised[2]. The former try to learn a function $f : D \rightarrow \mathcal{C}^*$ from a so-called training set of examples $(d_i, C_j) \in D \times \mathcal{C}^*$. The latter work without having seen correct category assignments in advance; their goal is to construct a categorization based only on D and a similarity measure φ.

The nature of supervised and unsupervised categorization algorithms determines their application area. For supervised categorization algorithms a category scheme as well as a considerable number of correctly assigned documents must be given in order to learn the function f. This fact limits the application area to collections that possess a rather static category structure, since a modification of $\mathcal{C}^*$ involves laborious human intervention: Training examples have to be reassigned to the new category scheme and the classifier f has to be relearned. Examples of collections with static topic structure include digital libraries of scientific articles, in which standardized category schemes like the ACM computer science classification scheme are used [8].

[1]In literature, a categorization $\mathcal{C}^*$ of D is often based on topic considerations. However, we generalize this view: the underlying category system may be organized by any meaningful criteria.

[2]For an introduction to machine learning, see e.g. [97, 35].

The ability of unsupervised algorithms to uncover category structures without prior knowledge is especially useful for post-retrieval categorization tasks in large, heterogeneous collections like intranets or the WWW: Due to their size and the user's different perspectives, these collections often lack commonly accepted category schemes. Moreover, unsupervised categorization in post-retrieval scenarios can adapt to a query's "zoom level": Depending on whether a query is general or specific, the range of relevant documents should be broad or focused, and accordingly, a categorization's granularity should adapt.

3.2 The Cluster Hypothesis

Browsing categories instead of result lists shall improve the retrieval performance, e.g. measured by the ratio of relevant documents within a query response. Van Rijsbergen formulated the underlying Cluster Hypothesis as follows: "Closely associated documents tend to be relevant to the same requests" [111]. However, the Cluster Hypothesis, if true, does not imply a concrete procedure for retrieval. Former studies experimented with retrieval strategies that varied in the following points.

(1) *Document base.* What is to be clustered? Alternatives are (a) the entire collection [153, 103] or (b) top-ranked documents that were retrieved for a query ("post-retrieval clustering") [24, 56, 132, 151].

(2) *Cluster selection.* What is shown to a user? Possibilities include (a) the documents that belong to the cluster that is closest to the query, e.g. measured by the query's distance to cluster centroids [153] (b) the documents that belong to the cluster whose centroid is closest to the top-ranked document with respect to the query [22] (c) the documents that belong to the top-k query-closest clusters [153, 155] (d) documents from all clusters grouped by cluster [132, 151].

(3) *Cluster presentation.* How are clusters presented to a user? Alternatives are (a) a single ranked list of documents that were retrieved from one or more clusters [153], (b) a ranked list grouped by cluster [56, 103] (c) a cluster overview for navigation combined with a list view [132, 151].

The mentioned points are pairwise orthogonal and may influence the retrieval performance to a great extent like the mentioned references show. E.g., clustering an entire document collection and selecting for presentation the documents belonging to the cluster that is closest to an input query has proven not to improve the retrieval quality in terms of precision [119, 153, 50]. As a consequence, Hearst and Pedersen proposed to cluster retrieval results instead of the whole collection ("Scatter/Gather") [56]. Their results on a large, heterogeneous document collection showed the great potential of the approach: retrieval from the query-closest cluster increased the retrieval precision by up to 85% in comparison to the retrieval precision from unclustered results. This observation does not support the Cluster Hypothesis directly, but a variant "with some assumptions revised" [56]:

> "[...] we do *not* assume that if two documents d_1 and d_2 are both relevant or nonrelevant for query q_A, they must also *both* be relevant or nonrelevant for query q_B. [...] In other words, because documents are very high-dimensional, the definition of nearest neighbors will change depending on which neighbors are on the street."

The next section will discuss current approaches for document clustering. It will present theoretical foundations, algorithms, and challenges of cluster analysis in the text domain.

3.3 Clustering Documents

The definition of a clustering $\mathcal{C} = \{C_1, \dots, C_k\}$ of a set D is—from a mathematical point of view—the same like the categorization definition given above: the sets C_i are (disjoint) subsets of D, and their union covers D. From a semantic point of view, $\mathcal{C}$ is called clustering if it was generated by a clustering algorithm. The goal of clustering techniques is to find clusterings that are close to the optimum categorization[3] $\mathcal{C}^*$. In general, cluster analysis concentrates on the following questions.

[3]Optimum categorizations are in our case human-made categorizations of the document set D. However, the definition of optimality depends on the underlying application scenario and should be oriented at the user's information need. In contrast to ranked retrieval, where the probability ranking principle defines optimality [112, 45], a formal, human-independent optimality definition for clusterings has not been stated so far.

(1) Are there clusters in D?

(2) How can the clusters be identified?

(3) How "good" are the found clusters?

Some statistics to answer question (1) have been proposed in the past [34]; a recent article that discusses this question in context of information retrieval can be found in [150]. However, our focus here is on the questions (2) and (3), which we will elaborate in this and the next section respectively.

3.3.1 Challenges

A clustering is considered having a good quality with respect to a similarity function $\varphi : D \times D \to \mathbf{R}$, if the averaged similarity between objects of the same cluster is "significantly higher" than the similarity between items from different clusters. Other desired properties include that clusters shall be of a "compact form" or resemble the "natural structure" of the item set [64, 71, 139].

A method to model the desired cluster properties is the specification of an objective function f_o that maps a clustering to a real number that quantifies the goodness of a clustering. However, since the number of possibilities to partition a set of n items into k partitions is roughly $k^n/k!$ [64], it is infeasible to exhaustively search the space of possible partitions for elements that maximize/minimize the value of f_o, even when the "right" k is known. Finding the right k is a challenge itself.

Apart from this general problem, document clustering using the standard vector space model must cope with the high dimensionality of the vectors that represent the documents. In particular, the curse of dimensionality and noise phenomenons must be considered when clustering high-dimensional data.

Curse of Dimensionality. The term "curse of dimensionality" was coined by Bellman [10] and circumscribes difficulties when analyzing high-dimensional data; particularly it relates to the exponential growth of hyper-volume when scaling dimensionality linearly. Consequently, clustering algorithms that rely on analyses of spatial density[4] must be designed carefully [57, 49]. According to Dasgupta, the curse of dimensionality expresses in two different facets [25]:

[4]An example for a class of such algorithms are grid-based clustering algorithms.

(1) *Visualization problems.* High-dimensional data are hard to visualize. Although there are dimension reduction techniques to embed high-dimensional data into lower-dimensional spaces while preserving similarity relations between data points as good as possible, caution is advised when inspecting the projected data visually or when analyzing the projected data with statistical tools.

    The first reason for caution is that a projection from a high-dimensional space is subject to a projection error, the so-called *stress* [77], which measures similarity differences between point-pairs in the original and the projected space. Depending on the application, the stress may influence the outcome to a greater or lesser extent. Our study in [131] illustrates this effect when clustering low-dimensional projections of high-dimensional document vectors.

    The second reason is that distribution properties of point sets may be altered when projected to a lower-dimensional space. For example, simple non-Gaussian distributions exist whose two-dimensional projections look Gaussian [32, 25].

(2) *Counter-intuitive pitfalls.* An example for such a pitfall is the following. When uniformly choosing points from an n-sphere at random, they almost always lie close to the surface of the sphere. This observation relates to the fact that uniformly drawn points from $[0, 1]^n$ tend to be equidistant as n rises. This and other related pitfalls that may be overlooked are described in [25].

In general, much of the technology developed for lower dimensions does not scale well to higher dimensions. Dasgupta mentions in this connection the partition of a vector space into Voronoi cells: m points in $\mathbf{R}^n$ will produce cells with $\Omega(m^n)$ faces.

Noise. Similarity values that are computed between two documents d_1 and d_2 are subject to noise. Clearly, even if d_1 and d_2 are not on the same topic there will be terms that are shared between both documents. Although the *idf* part of the $tf \cdot idf$ term weighting scheme reduces the weight for common terms within a

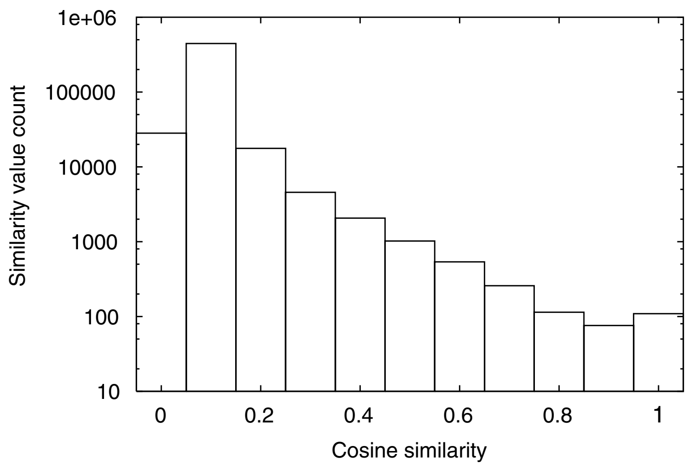


Figure 3.2: Distribution of similarity values for a document collection of 1000 news documents drawn uniformly at random from 10 categories. Observe the logarithmic scale.

collection, some random term matches are still possible that influence a similarity measure's value[5].

Figure 3.2 depicts a distribution of similarity values for a document collection of 1000 news documents drawn uniformly from 10 categories. The figure shows that there are many low similarity values due to random term matches. Clustering algorithms based on similarity graph density measures, for example, run the risk to sum up many of these values and add a remarkable error when judging cumulated similarity, e.g. between clusters.

3.3.2 Clustering Algorithms

Given a text corpus D, an optimum clustering $\mathcal{C}^*$ of D corresponds to its categorization as accomplished by a human editor. Except for trivial cases $\mathcal{C}^*$ cannot be found—the reasons for this are twofold: First, it is hard to quantify the properties of the "correct" categories in terms of the similarity function φ or the objective function f_o; secondly, the search space of possible clusterings is exponential in $|D|$ and a greedy strategy to find a global optimum is not at hand. I. e., cluster algorithms are highly developed heuristics to search a global optimum with respect to the outlined structure identification objectives.

[5]The probability can be bounded from below based on "Birthday Problem" estimates.

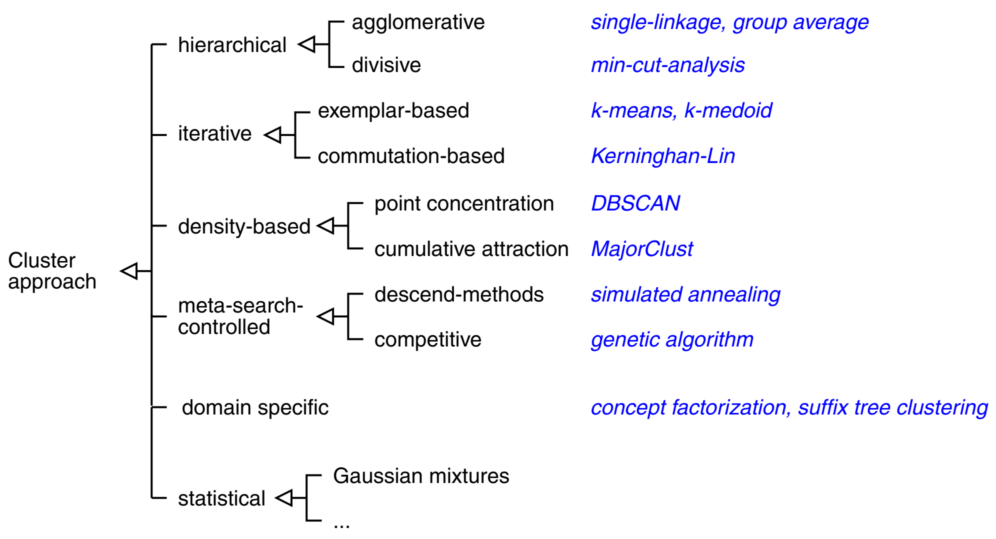


Figure 3.3: A taxonomy of clustering algorithms that are used for document categorization (adapted from [129]).

Various clustering heuristics and strategies have been proposed (cf. Figure 3.3), and it is hard to say which of the existing approaches is suited best for text categorization. The following paragraphs outline basic principles of the clustering algorithms that are used for document categorization. In this connection it is useful to consider the n elements in D as nodes of a weighted graph whose edge weights correspond to the similarity $\varphi(\mathbf{d_1}, \mathbf{d_2})$ of the respective documents.

Iterative Algorithms. Iterative algorithms strive for a successive improvement of an existing clustering and can be classified further into exemplar-based and commutation-based approaches. The former assume for each cluster a representative to which the objects become assigned according to their similarity. Iterative algorithms need information with regard to the expected cluster number, k. Well-known representatives are k-Means, k-Medoid, Kohonen, and Fuzzy-k-Means. The runtime of these methods is $\mathcal{O}(nkl)$, where l designates the number of iterations to achieve convergence [64, 88, 71, 74, 160, 52, 53, 75, 62].

Hierarchical Algorithms. Algorithms of this type create a tree of node subsets by successively dividing or merging the graph's nodes. In order to obtain a unique clustering, a second step is necessary that prunes this tree at adequate places. Agglomerative hierarchical algorithms start with each vertex being its own cluster and union clusters iteratively. For divisive algorithms on the other hand, the entire graph initially forms one single cluster which is successively subdivided. Representatives are k-nearest-neighbor, linkage, Ward, minimum-spanning-tree, or min-cut methods. Usually, these methods construct a complete similarity graph, which results in $\mathcal{O}(n^2)$ runtime [38, 40, 125, 67, 84, 158, 162, 69].

Density-based Algorithms. Density-based algorithms try to separate a similarity graph into subgraphs of high connectivity values. In the ideal case they can determine the cluster number k automatically and detect clusters of arbitrary shape and size. Representatives are DBscan, MajorClust, or Chameleon. The runtime of these algorithms cannot be stated uniquely since it depends on diverse constraints. For non-geometrical data it is in the magnitude of hierarchical algorithms, $\mathcal{O}(n^2)$ [139, 37, 69, 130, 131].

Meta-Search Algorithms. Meta-search algorithms treat clustering as an optimization problem where a given goal criterion is to be minimized or maximized. Though this approach offers maximum flexibility only less can be stated respecting its runtime. Representatives for meta-search driven cluster detection may be realized by genetic algorithms, simulated annealing, or a two-phase greedy strategy [9, 118, 117, 108, 41, 73].

Domain Specific Algorithms. Various clustering algorithms have been proposed for high-dimensional data or especially for the text domain. The goal of the latter is often to identify concepts, which are in turn the basis for grouping documents. Techniques that are employed in this connection include projections of the high-dimensional data to lower dimensions each of which representing a concept, e.g. low-rank approximations of the term-document matrix. Other approaches try to identify concepts directly with frequency or subsumption analyses. Some recent algorithms are given in [163, 159, 70, 31, 49].

Statistical Parameter Estimation Algorithms. This class of algorithms regards a population of points in $\mathbf{R}^n$ as a sample that has been generated by a mixture model of k n-dimensional density functions $\delta_1, \ldots, \delta_k$ with different parameters, where the density functions have been combined according to a weighting scheme $w_1, \ldots, w_k$ with $\sum_i w_i = 1$. The goal is to estimate the parameter set of each density function δ_i, each of which being regarded as generator for a particular cluster. The probability that a point is generated by δ_i can then be calculated based on the density estimates, and the cluster numbers can be assigned to the data points according to this probability. An up-to-date compilation of such algorithms can be found in [25, 1].

3.4 On the Validity of Document Clusterings

Even when a document model along with a similarity measure is determined for a particular categorization task, a muchness of clusterings can be generated by clustering algorithms. The quality of unsupervised categorizations varies according to the employed cluster algorithms, especially in combination with their parameter settings. Since hundreds of clusterings of a mid-size document collection can be generated within one second CPU time, the difficulty with clustering is not the time factor but the selection of a valid clustering, say, a clustering that reflects the human idea of categorization best. Jain and Dubes [64] comment this difficulty as follows:

> "The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage."

This is where cluster validity measures come into play; their task is to decide which of the generated clusterings is best, either closest to a reference categorization that is regarded as optimum, or well developed with respect to some of the clustering's structural properties. The former are called external measures, the latter internal measures (cf. Figure 3.4). Measures that are capable to judge which clustering is best among a set of clusterings are sometimes referred to as relative measures.

In the following, a more detailed taxonomy of validity indices is introduced, and the usage of particular measures is discussed. In particular, we introduce a new internal cluster validity measure, called expected density, that proved to consistently outperform other traditional internal measures in the document clustering domain [138]. Experimental analyses conclude the section.

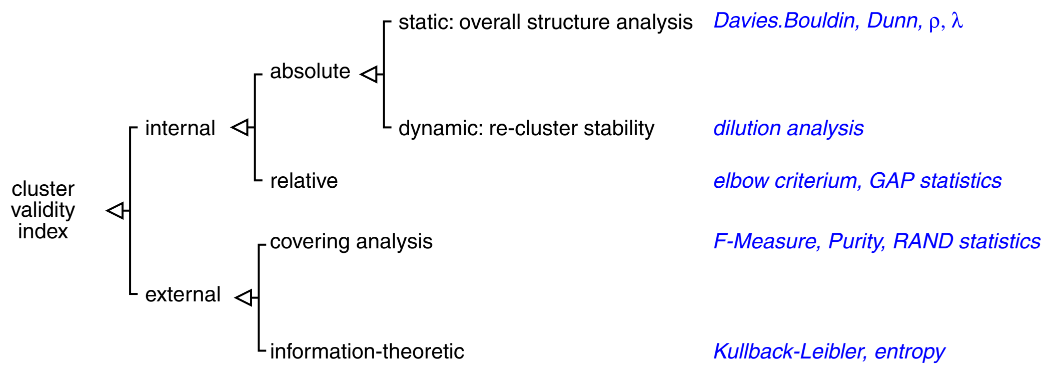

cluster validity index
internal
external
absolute
relative
static: overall structure analysis
dynamic: re-cluster stability
covering analysis
information-theoretic
Davies.Bouldin, Dunn, ρ, λ
dilution analysis
elbow criterium, GAP statistics
F-Measure, Purity, RAND statistics
Kullback-Leibler, entropy

Figure 3.4: Taxonomy of cluster validity indices.

3.4.1 External Cluster Validity Measures

External measures are subject to quantify the congruence between a given clustering and a reference categorization. In document clustering, the latter is usually a document collection that has been categorized by a human editor. In particular, external measures can be subdivided into information theoretic and covering analysis approaches (cf. Figure 3.4). External measures are widely used for measuring the performance of a clustering or classification algorithm against a benchmark collection.

The F-Measure. The F-Measure combines the precision and recall measures from information retrieval [111]. While precision quantifies the "purity" of a cluster with respect to a reference class, the recall value captures the percentage of data objects from a reference class that are contained in a particular cluster.

Let D represent a set of documents and let $\mathcal{C} = \{C_1, \dots, C_k\}$ be a clustering of D. Moreover, let $\mathcal{C}^* = \{C_1^*, \dots, C_l^*\}$ designate the human reference categorization.

Then the recall of cluster j with respect to class i, $rec(i,j)$, is defined as $|C_j \cap C_i^*|/|C_i^*|$. The precision of cluster j with respect to class i, $prec(i,j)$, is defined as $|C_j \cap C_i^*|/|C_j|$. The F-Measure is the harmonic mean of precision and recall:

$$F_{i,j} = \frac{1}{\frac{1}{2}\left(\frac{1}{prec(i,j)} + \frac{1}{rec(i,j)}\right)} = \frac{2 \cdot prec(i,j) \cdot rec(i,j)}{prec(i,j) + rec(i,j)}$$

Based on this formula, the overall F-Measure of a clustering (micro-averaged) is:

$$F = \sum_{i=1}^{l} \frac{|C_i^*|}{|D|} \cdot \max_{j=1,\dots,k} \{F_{i,j}\}$$

An alternative way to compute F uses macro-averaging: the maxima are not weighted according to class size, but all classes are weighted equally, i.e.

$$F = \frac{1}{l} \sum_{i=1}^{l} \max_{j=1,\ldots,k} \{F_{i,j}\}$$

In other words, macro averaging treats classes equally, while micro averaging treats documents equally. Unless stated otherwise, we will employ micro-averaging as it considers class importance.

Note that parallel maximization of precision and recall is difficult; the F-Measure quantifies to what extent this objective is met for a particular clustering. A perfect fit between a generated clustering and a human reference categorization leads to an F-Measure score of 1, which is the maximum value for this measure.

Entropy. Entropy is an information theoretic measure that quantifies the uncertainty of an information source S, which emits symbols $S_1, \ldots, S_k$ according to the probabilities $P(S_1), \ldots, P(S_k)$. The entropy of the information source is defined [124] as

$$H(S) = -\sum_{i=1}^{k} P(S_i) \cdot \log_2(P(S_i))$$

The connection between entropy and validity measures is as follows. Imagine each cluster of a given clustering acting as information source. As impure clusters comprise data from different "true" classes, one can imagine that a cluster emits class numbers from the underlying reference categorization. Whenever a class number is emitted from a cluster, its probability corresponds to the fraction of the number of items from its class within the cluster, and the total number of items within the emitting cluster. Obviously, an impure cluster, which contains data from different classes, represents an uncertain information source, and thus results in a high entropy value.

Let D represent the set of documents, and let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a clustering of D. Moreover, let $\mathcal{C}^* = \{C_1^*, \ldots, C_l^*\}$ designate the human reference categorization.Then the entropy of cluster C_i with respect to $\mathcal{C}^*$ is defined as

$$H(C_j) = -\sum_{|C_j \cap C_i^*| \neq 0} \frac{|C_j \cap C_i^*|}{|C_j|} \cdot \log_2 \left(\frac{|C_j \cap C_i^*|}{|C_j|} \right)$$

The entropy of the entire clustering $\mathcal{C}$ with respect to $\mathcal{C}^*$ is the sum of the cluster-wise entropies, which are weighted according to cluster size (micro-averaged):

$$H(\mathcal{C}) = \sum_{C_j \in \mathcal{C}} \frac{|C_j|}{|D|} H(C_j)$$

Observe that a perfect clustering has an entropy of 0. However, the pathological case in which each data object is assigned to its own cluster also scores with 0. The reason for this behaviour is that entropy mathematically transforms the *precision* values of the clusters.

Pair-Analysis-based Figures of Merit. Another class of external measures analyzes to which extent document pairs share the same cluster and the same underlying class, and quantifies the congruence. Again, let $D = \{d_1, \dots, d_n\}$ be the set of documents, let $\mathcal{C} = \{C_1, \dots, C_k\}$ be a clustering of D, and let $\mathcal{C}^* = \{C_1^*, \dots, C_l^*\}$ designate the human reference categorization. Let $t_{\mathcal{C}} : \mathbf{N} \times \mathbf{N} \rightarrow \{0, 1\}$ be an indicator function with $t_{\mathcal{C}}(i, j) = 1$ if d_i and d_j share the same cluster in $\mathcal{C}$, and 0 otherwise. Likewise, $t_{\mathcal{C}^*}(i, j) = 1$ if d_i and d_j share the same class in $\mathcal{C}^*$, and 0 otherwise. Based on these indicator functions, the Rand statistic [109] of a clustering $\mathcal{C}$ is defined as

$$R(\mathcal{C}) = \frac{1}{n(n-1)/2} \sum_{i<=j} \delta_{t_{\mathcal{C}}(i,j), t_{\mathcal{C}^*}(i,j)}$$

where $\delta_{x,y}$ is the Kronecker symbol, i.e. $\delta_{x,y} = 1$ if $x = y$, and 0 otherwise.

In other words, the Rand statistic counts the pairs of documents that lie both in the same or both in different clusters in $\mathcal{C}$ and $\mathcal{C}^*$. The resulting sum is normalized with the total number of pairs. Consequently, the values stem from the interval $[0, 1]$ with 1 indicating a perfect match between $\mathcal{C}$ and $\mathcal{C}^*$.

There are also other statistics that rely on the given indicator functions [64]. They include the Fowlkes and Mallows statistic [42], the Jaccard statistic [64], and the Γ statistic [60].

3.4.2 Internal Cluster Validity Measures

In contrast to external measures, internal measures quantify the quality of a clustering based on its structural properties only, i. e. , without comparing it to the “right” categorization. They can further be split up into relative and absolute measures (cf. Figure 3.4), whereas the task of relative measures is to identify the best in a set of clusterings, and the task of absolute measures is to judge the quality of a single clustering by means of a real number. Although there are measures that can only be employed relatively since they require a set of clusterings as input, it should be noted that absolute measures can also be used relatively, when comparing their values for different clusterings[6].

The most popular internal measures are absolute measures that focus on static structural properties like compactness and well-separateness, measured by within-cluster-scatter, between-cluster-distance, etc. Alternatively, structural properties may be assessed in a dynamic way employing re-clustering techniques, which aim to assess the stability of a clustering. In the following, some well-known as well as new validity measures will be introduced.

The Dunn Index Family. Dunn Indices [36] form one of the most popular classes of internal validity measures. Dating back to 1973, Dunn's measure was one of the first to be investigated, and Bezdek generalized Dunn's family of indices by abstracting cluster diameter and cluster distance measures as follows [13].

Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a clustering of a document collection D, $\delta : \mathcal{C} \times \mathcal{C} \to \mathbf{R}$ be a cluster to cluster distance measure, and $\Delta : \mathcal{C} \to \mathbf{R}$ be a cluster diameter measure. Then all measures I of the form

$$I(\mathcal{C}) = \frac{\min_{i \neq j}\{\delta(C_i, C_j)\}}{\max_{1 \leq l \leq k}\{\Delta(C_l)\}}$$

are called Dunn indices. Originally, Dunn used

$$\delta(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y) \quad \text{and}$$

$$\Delta(C_i) = \max_{x, y \in C_i} d(x, y)$$

where $d : D \times D \to \mathbf{R}$ is a function that measures the distance between objects of D. Using these functions, the measure assigns high values to clusterings with compact and very well separated clusters like shown in Figure 3.5. Here, the maximum diameter is relatively low and the minimum distance between two clusters is relatively large. As a consequence, the clustering is ranked high by

[6]It is assumed that the values of absolute measures relate monotonously to cluster quality.

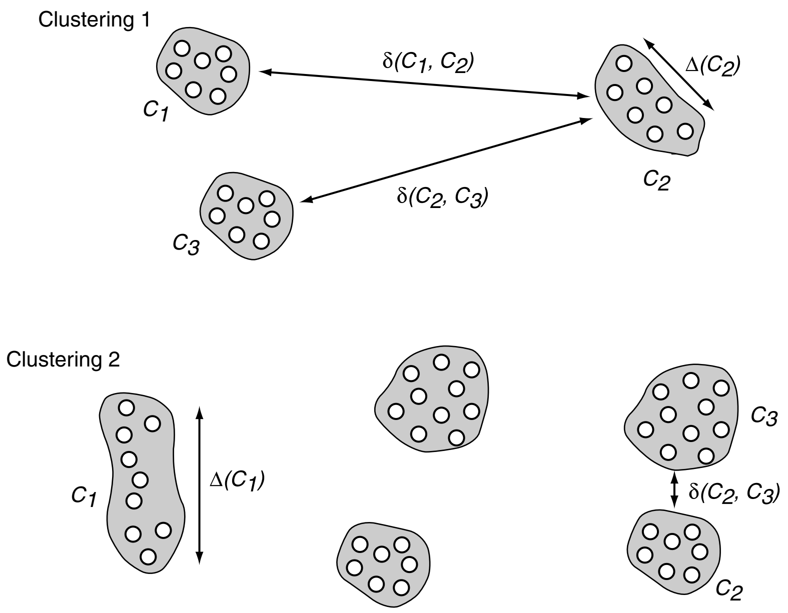


Figure 3.5: Clustering 1 contains a clustering with compact and well separated clusters. The maximum diameter is relatively low and the distance of the two closest clusters is relatively large. In this case the original Dunn index returns a high score for the clustering. Clustering 2 shows two undesired properties from the perspective of the Dunn Index: the left cluster's large diameter as well as the relatively small distance between C_2 and C_3 are responsible for the index returning a low value, even though the clustering fits the structure well. If C_2 and C_3 are merged to one cluster, the Dunn Index would falsely return a better value.

the Dunn index. However, Bezdek recognized that the index is noise sensitive. Moreover, clusterings with arbitrarily shaped clusters can cause problems for the Dunn index (see Figure 3.5 bottom): Even though the clustering in the example is good, the large diameter of C_1 along with the small distance between C_2 and C_3 leads to a low value of the Dunn index. Bezdek experienced that the combination of

$$\delta(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i, y \in C_j} d(x, y) \quad \text{and}$$

$$\Delta(C_i) = 2\Big(\frac{\sum_{x \in C_i} d(x, c_i)}{|C_i|}\Big)$$

give reliable results for data with normal mixture distributions [12]. Here, c_i denotes the centroid of cluster C_i. Note that a maximization of I is desired.

The Davies-Bouldin Index. Davies and Bouldin proposed the following measure that is known as Davies-Bouldin Index [26]. It is a function that combines within-cluster scatter and between-cluster separation as follows. Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a clustering of a document collection D. Then the Davies-Bouldin Index $DB : \mathcal{C} \to \mathbf{R}$ is defined as

$$DB(\mathcal{C}) = \frac{1}{k} \cdot \sum_{i=1}^{k} R_i(\mathcal{C}), \quad \text{with}$$

$$R_i(\mathcal{C}) = \max_{\substack{j=1,\ldots,n,\\ i \neq j}} R_{ij}(\mathcal{C}) \quad \text{and} \quad R_{ij}(\mathcal{C}) = \frac{(s(C_i) + s(C_j))}{\delta(C_i, C_j)},$$

where $s : \mathcal{C} \to \mathbf{R}$ measures the scatter within a cluster, and $\delta : \mathcal{C} \times \mathcal{C} \to \mathbf{R}$ is a cluster to cluster distance measure.

Given the centroids c_i of the clusters C_i, a typical scatter measure is $s(C_i) = \frac{1}{|C_i|} \sum_{x \in C_i} ||x - c_i||$, and a typical cluster to cluster distance measure is the distance between the centroids, $||c_i - c_j||$. Because a low scatter and a high distance between clusters lead to low values of R_{ij}, a minimization of DB is desired.

Cluster Validity by Resampling. The application of resampling for cluster validity was introduced in 1987 by Jain and Moreau [63]. Resampling bases on the idea to compare a given clustering $\mathcal{C}$ to a clustering of a randomly chosen subset from the underlying document set. If the subsets' clusterings differ significantly from $\mathcal{C}$, then $\mathcal{C}$ is instable. In this case, the clustering is bad or the underlying data does not contain a structure. In general, the resampling procedure is carried out in the following steps.

(1) Several resamples $D_j \subset D$ with $|D_j| < |D|$ are randomly chosen from the original data set. The fraction $\frac{|D_j|}{|D|}$ is called dilution factor [86].

(2) A clustering algorithm is applied to each D_j, yielding to re-clusterings $\mathcal{R}_j$. Here, the cluster algorithm's parameters must be identical to those used for generating the clustering $\mathcal{C}$.

(3) Similarities $\psi(\mathcal{R}_i, \mathcal{C})$ between the original clustering and the re-clusterings are calculated and averaged. For this task, external validity measures can be employed, with $\mathcal{C}$ acting as reference categorization.

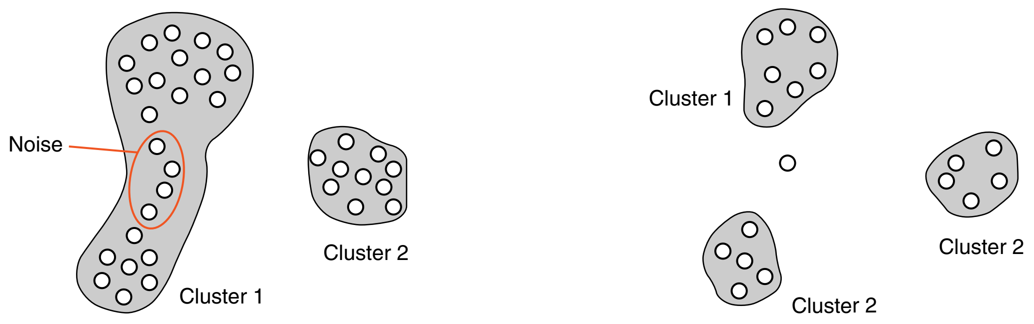


Figure 3.6: Illustration of resampling. On the left hand side a clustering is shown, which was generated by a cluster algorithm that was mislead by noise. On the right hand side, a clustering of a resampling can be seen, which identifies the regions in which stable clusters reside.

The average difference between the original clustering and the resample clusterings is an indicator for the quality of $\mathcal{C}$. An intuition for this interpretation is that noise in the data set may loose influence on the clustering algorithm when resamples are drawn. Probably, weak clusters that are influenced by noise will break into parts during the re-clustering step. Consider Figure 3.6 in this connection: After resampling, enough points of stable clusters remain so that the clustering algorithm is able to detect them in several resamplings.

Note, however, that values of a clustering similarity measure ψ should only be compared or averaged for re-clusterings that were generated using the same dilution factor, as upper/lower bounds for ψ are determined by the number of points in D_i. For example, the F-Measure-values for re-clusterings that were generated using a dilution factor of 0.5 will likely be smaller than those generated with a dilution factor of 0.9 if the same clustering algorithm with the same parameters is employed.

The Elbow Criterion. The elbow criterion is a relative validity measure, i. e., it decides which clustering is best within a set of clusterings. Let $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_{p_1}, \ldots, \mathcal{C}_{p_k}\}$ be a set of clusterings that has been generated with the same clustering algorithm but with different parameter values $p_1, \ldots, p_k$, and let $e : \boldsymbol{\mathcal{C}} \rightarrow \mathbf{R}$ be an error function on the clusterings such as the sum over all clusters of within-cluster scatter with respect to the cluster centroid. Let $\{(p_i, e(\mathcal{C}_{p_i})) \mid i \leq k\}$ be a set of points forming an error curve; without loss of generality, assume that the $(p_i, e(\mathcal{C}_{p_i}))$ are ordered by descending $e(\mathcal{C}_{p_i})$ values. If point $(p_i, e(\mathcal{C}_{p_i}))$ experiences the maximum error drop with respect to its predecessor, the parameter p_i

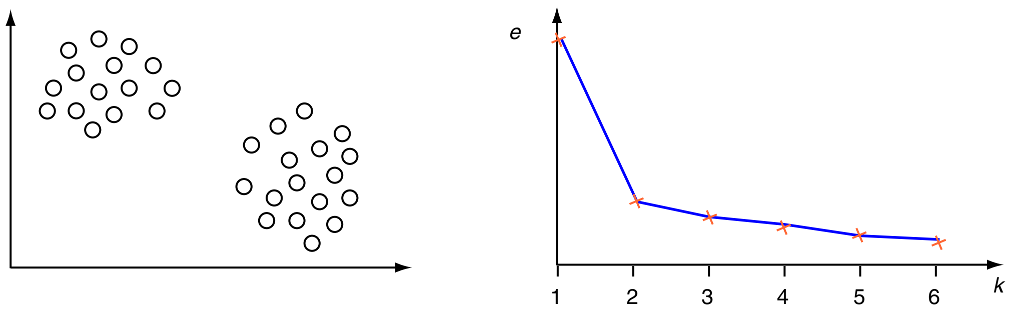


Figure 3.7: On the left, the figure shows some points in a 2-dimensional space, forming two clusters. On the right, an error function e is depicted for k-Means generated clusterings of the points, depending on the specified number of clusters, k. The error drop for $k = 2$ is maximum, indicating that $k = 2$ is a good parameter for the underlying data.

is considered as good choice for the clustering algorithm.

As an example, confer Figure 3.7. The clustering algorithm k-means tends to reduce the overall variance of a generated clustering on the same data when k increases, and when k reaches $|D|$, the variance disappears since each cluster tends to contains only a single point. The depicted error function has its maximum variance drop in the second point, indicating that $k = 2$ is a good parameter for the underlying data.

An advanced approach is the GAP statistic, which is an error-tolerant and normalized variant of the elbow criterion [144].

The Λ-Measure. A document collection can be considered as a weighted graph $G = \langle V, E, w\rangle$ with node set V, edge set E, and weight function $w : E \rightarrow [0, 1]$ where V represents the documents, and w defines the similarities between two documents. The Λ-measure computes the weighted partial connectivity of $G = \langle V, E, w\rangle$, which is defined as follows [139].

Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a clustering of the nodes V of a weighted graph $G = \langle V, E, w\rangle$.

$$\Lambda(\mathcal{C}) := \sum_{i=1}^{k} |C_i| \cdot \lambda_i,$$

where λ_i designates the weighted edge connectivity of the induced subgraph $G(C_i) = \langle V_i, E_i\rangle$. λ_i is defined as $\min_{E'} \sum_{\{u,v\} \in E'} w(u, v)$ where $E' \subset E$ and $G'_i = \langle V_i, E_i \setminus E'\rangle$ is not connected. λ_i is also designated as the capacity of a minimum cut of $G(C_i)$.

Measure of Expected Density $\overline{\rho}$. A graph $G = \langle V, E, w\rangle$ is called sparse if $|E| = \mathcal{O}(|V|)$; it is called dense if $|E| = \mathcal{O}(|V|^2)$. Put another way, we can compute the density θ of a graph from the equation $|E| = |V|^\theta$. With $w(G) := |V| + \sum_{e\in E} w(e)$ denoting the weight of G[7],this relation extends naturally to weighted graphs:

$$w(G) = |V|^\theta \quad \Leftrightarrow \quad \theta = \frac{\ln\left(w(G)\right)}{\ln\left(|V|\right)}$$

Obviously, θ can be used to compare the density of each induced subgraph $G' = \langle V', E', w'\rangle$ of G to the density of the entire graph G: G' is sparse (dense) compared to G if the quotient $w(G')/(|V'|^\theta)$ is smaller (larger) than 1. This consideration is the key idea behind the following definition of a clustering's expected density $\overline{\rho}$.

Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a clustering of a weighted graph $G = \langle V, E, w\rangle$, and let $G_i = \langle V_i, E_i, w_i\rangle$ be the induced subgraph of G with respect to cluster C_i. Then the expected density of a clustering $\mathcal{C}$ is defined as follows.

$$\overline{\rho}(\mathcal{C}) = \sum_{i=1}^{k} \frac{|V_i|}{|V|} \cdot \frac{w(G_i)}{|V_i|^\theta}, \quad \text{where} \quad |V|^\theta = w(G)$$

Since the edge weights resemble the similarity of the objects which are represented by V, a higher value of $\overline{\rho}$ indicates a better clustering.

Remarks. The Dunn index, the Davies-Bouldin index, and the elbow criterion are related in that they have a geometric (typically centroidic) view on the clustering. The measures work well if the underlying data contains clusters of spherical form, but they are unreliable if this condition does not hold. Λ as well as $\overline{\rho}$ interpret a data set as a weighted similarity graph; they analyze the graph's edge density distribution to judge the quality of a clustering.

3.4.3 Statistical Hypothesis Testing

An internal cluster validity index quantifies the quality of a given clustering with regard to certain structural properties. But even though it is obvious whether

[7]The corrective summand $|V|$ in $w(G)$ assures that $w(G) \geq |V|$, which in turn assures that $\theta \geq 1$, and as a consequence, that the desired property $|V_i|^\theta + |V_j|^\theta \leq |V_i \cup V_j|^\theta$ holds. This correction is necessary for graphs with small edge weights in $[0, 1]$.

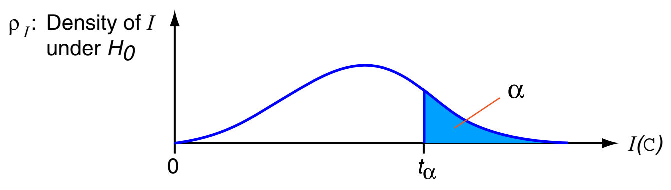


Figure 3.8: Estimated density function ρ_I of I under H_0. The critical value t_α of I is marked for significance level α.

large or small values of a particular index indicate better or worse clusterings, it is usually unknown which absolute values qualify a clustering being good or bad. In some settings a value of 0.7 may be sufficiently large to indicate a good clustering, in other settings a value of 0.8 is not large enough to accept a clustering. This dilemma can be addressed by statistical test theory [64].

A statistical test indicates in our context whether a clustering $\mathcal{C}$ fits the data unusually well. Here, "unusually well" means that $\mathcal{C}$ fits the data better than a randomly generated clustering. Given that each document of D is labeled according to cluster membership, a statistical test starts with the hypothesis

H_0 : All permutations of the labels on the n documents are equally likely.

Let I be a real-valued (internal) cluster validity index for which a greater value indicates a better clustering quality. Assumed that the distribution or density function ρ_I of the values of I under the null hypothesis H_0 is known (as depicted in Figure 3.8), the probabilities $P(E \mid H_0)$ could be determined, where E is the event "$I(\mathcal{C}) \leq t$" for some t. For a given significance level $0 \leq \alpha \leq 1$, one is interested in a threshold t_α such that $P(I(\mathcal{C}) \leq t_\alpha \mid H_0) = \alpha$. In this sense, t_α is a critical value. Formally, t_α is determined by solving the equation:

$$\int_{t_\alpha}^{\infty} \rho_I(t)dt = \alpha$$

and the decision rule

"IF $I(\mathcal{C}) > t_\alpha$ THEN reject H_0 at significance level α."

applies, assuming that a greater value of I indicates a better clustering.

If the null hypothesis is rejected, then the clustering $\mathcal{C}$ is assumed to have a non-random structure at significance level α. However, even though the null hypothesis is not supported by the test, it could still be true. But the probability of mistakenly rejecting H_0 is bounded by α, which in turn is controlled by the tester. α is usually set to 0.05 or 0.01 to ensure a small risk of making a mistake.

Another problem is that rejecting the hypothesis H_0 does not mean that the found clusters in $\mathcal{C}$ are meaningful—the test can only give evidence with respect to H_0. A test against a stronger H_1 hypothesis like "there are two clusters in D" would be desirable.

The main problem with this test is that the distribution of I under H_0 is unknown and must be estimated. The following Monte Carlo method can be applied to the problem [64]:

(1) Create a clustering $\mathcal{C}$ by assigning cluster labels $1, \ldots, k$ randomly to the elements of D. This procedure reflects the random label hypothesis H_0.

(2) Compute the cluster validity index $I(\mathcal{C})$, and count it as a match for a corresponding interval.

(3) Repeat steps one and two m times for a sufficiently large m.

(4) Derive an approximation of the density function ρ_I from the collected data.

The disadvantage of a statistical test is its high computational cost. Note that m must grow adequately when α reaches 0 to ensure the validity of the test. Moreover, the Monte Carlo estimation of I's distribution does not only depend on the null hypothesis H_0 and the generated clusterings, but also on the underlying document set D.

3.4.4 Experimental Evaluation

The experiments have been conducted with samples of RCV1, short hand for "Reuters Corpus Volume 1" [115]. RCV1 is a document collection that was published by the Reuters Corporation for research purposes. It contains over 800,000 documents each of which consisting of a few hundred up to several thousands words. The documents are enriched by meta information like category (also called topic), geographic region, or industry sector. There are 103 different categories, which are arranged within a hierarchy of the four top level categories "Corporate/Industrial", "Economics", "Government/Social", and "Markets". Each of

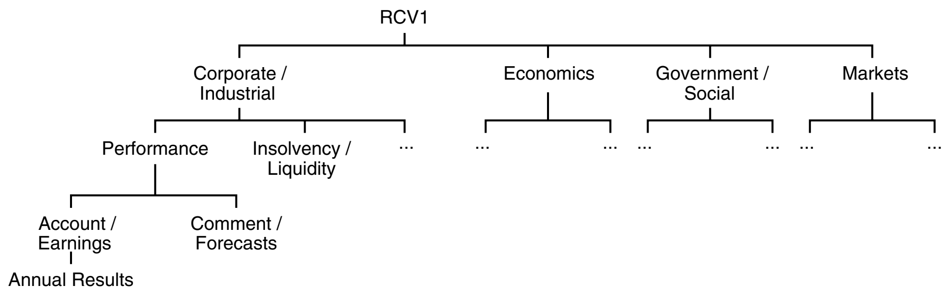


Figure 3.9: A part of the topic structure of RCV1.

the top level categories defines the root of a tree of sub-categories, where every child node fine grains the information given by its parent (cf. Figure 3.9). Note that a document d can be assigned to several categories $c_1, \dots, c_p$, and that all ancestor categories of a category c_i are assigned to d as well.

For our experiments, we considered two documents d_1, d_2 as belonging to the same category c_s if they share both the same top level category c_t and the same most specific category c_s. Moreover, we constructed the test sets in such a way that there is no document d_1 whose most specific category c_s is an ancestor of the most specific category of some other document d_2.

The number of categories in our test data varies from three to six. For each category, between 100 and 300 documents were drawn randomly from the entire category. The data sets had different sizes and class numbers; we investigated uniformly as well as non-uniformly distributed category sizes. Table 3.1 gives an overview of the constructed data sets.

The preprocessing of the documents included parsing of text body and title, stop word removal according to standard stop word lists, the application of Porter's stemming algorithm [106], and indexing according to term frequency. We used the standard cosine similarity measure to capture the similarities between documents.

Three analyses were conducted on each test data set to evaluate the performance of the aforementioned indices. The three analyses along with their results are described in the next three subsections.

Consistence Analysis. Since we know the reference categorization $\mathcal{C}^*$ which was provided by a human editor, we can use it to generate artificial clusterings $\mathcal{C}_1, \dots, \mathcal{C}_n$ that are to a greater or lesser extent modifications of C^*. The

	DS1	DS2	DS3	DS4	DS5	DS6	DS7
# categories	3	3	4	3	5	5	6
# documents	300	600	400	450	450	800	900
unif. distributed	yes	yes	yes	yes	yes	no	no

Table 3.1: Overview of the constructed data sets.

F-Measure values for $\mathcal{C}_1, \ldots, \mathcal{C}_n$ will measure the degree of congruence for the modified sets with respect to $\mathcal{C}^*$. Assuming that the modified categorizations represent erroneous clusterings, the value of a validation index for $\mathcal{C}_1, \ldots, \mathcal{C}_n$ should be worse than for $\mathcal{C}^*$. Even more can be expected: For $\mathcal{C}_1, \ldots, \mathcal{C}_n$, the values of a subjective validation index should relate to the values of the F-Measure monotonically.

To derive an artificial clustering $\mathcal{C}_i$ of $\mathcal{C}^*$, we repeatedly chose two distinct clusters of $\mathcal{C}^*$ and interchanged randomly chosen subsets of documents pairwise between the clusters. Note that the size of the interchanged subsets controls the degree of congruence between $\mathcal{C}_i$ and $\mathcal{C}^*$. We varied the sizes of the subsets between 1 document and 50% of the documents within a cluster.

Figure 3.10 shows the resulting scatter plots for the artificial clusterings that we derived from the reference categorization of DS5 and evaluated with the data set DS5. We measured the F-Measure value (y-axis) and the validity index value (x-axis) for each clustering. For the sake of better readability, we changed the sign of the Davies-Bouldin Index, which is the only one to be minimized—this way, the plots are directly comparable. Assuming that a greater index value constitutes a better clustering, an ideal index would show points on a curve that starts in the lower left corner and grows monotonically up to the upper right corner.

	DS1	DS2	DS3	DS4	DS5	DS6	DS7	average
Dunn	0.45	0.89	0.42	0.66	0.59	0.77	0.75	0.65
D.-B.	0.95	0.98	0.93	0.84	0.91	0.86	0.92	0.91
$\overline{\rho}$	0.96	0.99	0.94	0.97	0.98	0.94	0.97	0.96

Table 3.2: The table shows for each index the Spearman rank correlation coefficient of the *artificial clusterings* for the investigated data sets. The number of the underlying clusterings is 30.

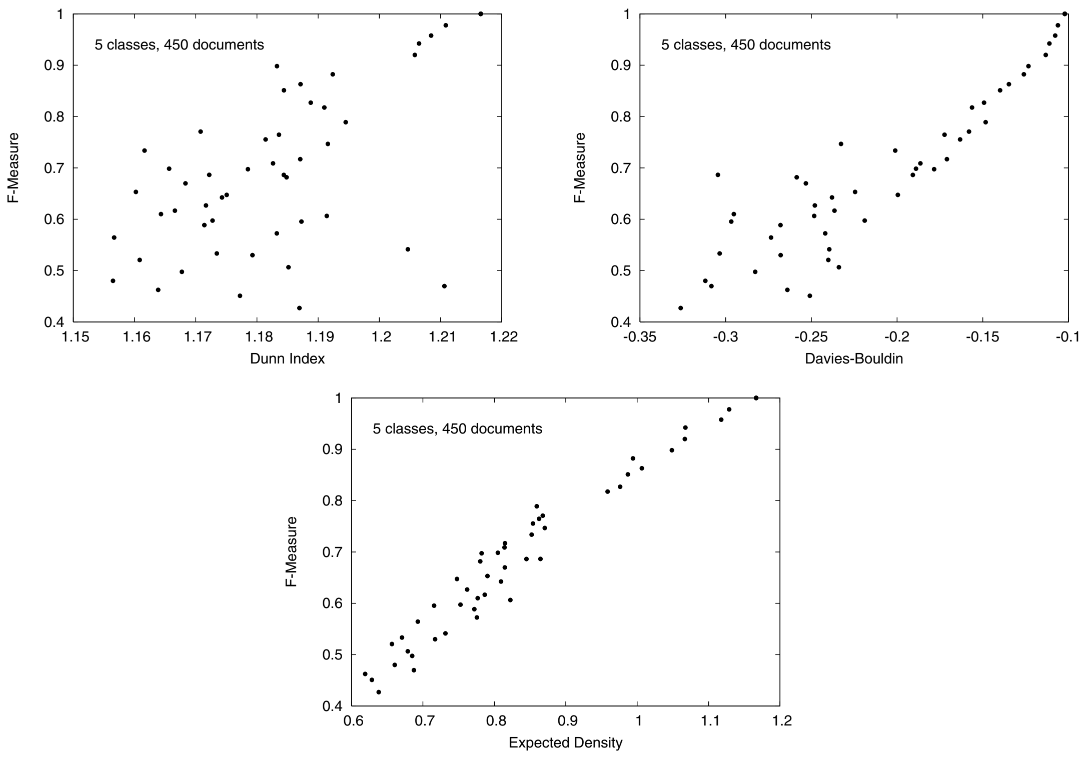


Figure 3.10: Correlation of the F-Measure with (a) Dunn's Index (top left), (b) Davies-Bouldin-Index (top right) and (c) Expected Density (bottom) for the *artificial clusterings* on DS5.

The extent to which this property is resembled by an index can be quantified with Spearman's rank correlation coefficient. We determined for 30 clusterings their rank according to both, F-Measure and index value, and we quantified the correlation of these rankings as depicted in Table 3.2. Since the critical value for Spearman's rank correlation coefficient for 30 items is 0.43 at the significance level $\alpha = 0.01$, it can be concluded that the indices perform well on this test on all data sets (with one exception).

Analysis with Genuine Clusterings. We are normally faced with clusterings which are not artificially constructed but stem from a document categorization system that uses different clustering algorithms. For the experiments reported below we employed hierarchical, iterative, and density-based algorithms. Moreover, for each of these algorithms different thresholds, agglomeration levels, cluster numbers, etc. were tried. We measured the F-Measure values and corresponding validity index value for each clustering, and, in particular, for the reference

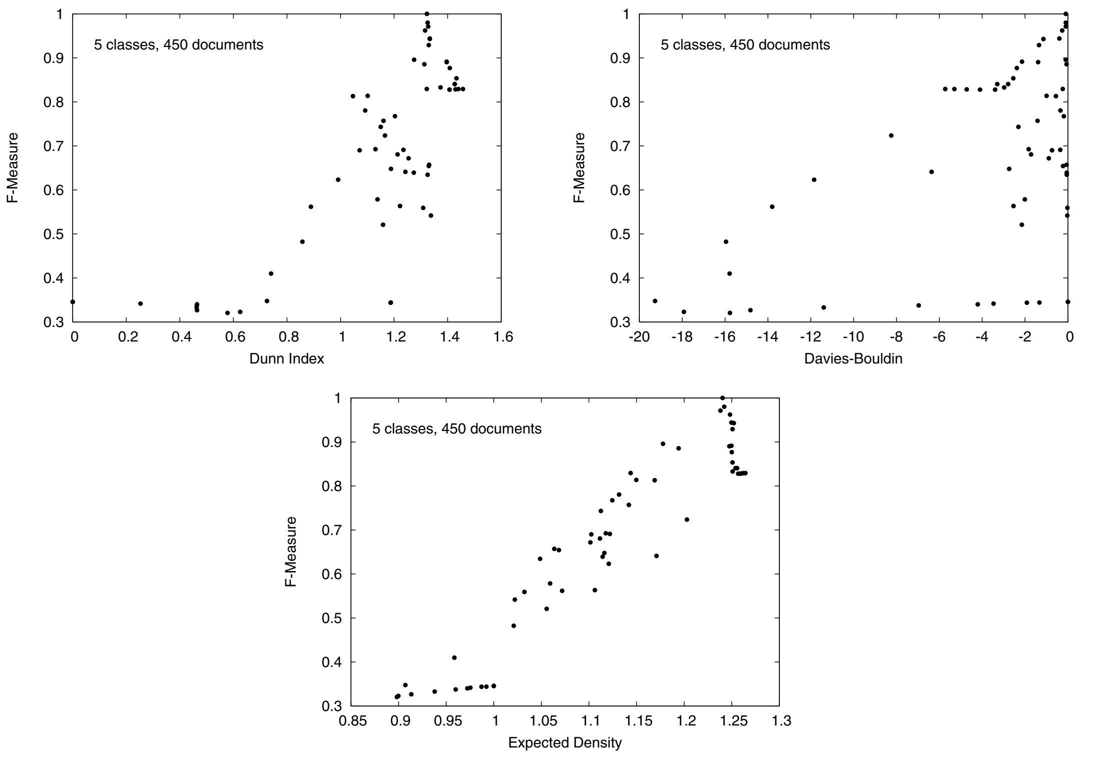


Figure 3.11: Correlation of the F-Measure and the Dunn Index (top left), Davies-Bouldin-Index (top right) and Expected Density (bottom) for *genuine clusterings* on DS5.

categorization $\mathcal{C}^*$ that can be identified by its F-Measure value of 1.

Figure 3.11 shows representative scatter plots for the investigated validity indices for the same data set that was used for the consistency analysis (DS5) with the difference that the clusterings were generated using different clustering algorithms. Again, we changed the sign of the Davies-Bouldin Index. Like above, we computed the rank correlations according to Spearman; these values can be found in Table 3.3.

	DS1	DS2	DS3	DS4	DS5	DS6	DS7	average
Dunn	0.69	0.64	0.78	0.64	0.71	0.68	0.76	0.70
D.-B.	0.58	0.53	0.49	0.53	0.38	0.44	0.40	0.48
$\overline{\rho}$	0.79	0.76	0.86	0.90	0.90	0.78	0.71	0.81

Table 3.3: The table shows for each index the Spearman rank correlation coefficient of the *genuine clusterings* for the investigated data sets. The number of the underlying clusterings is 30.

Note that the Davies-Bouldin Index, which works well for the synthetic data sets, gets mislead by the genuine clusterings generated by our clustering algorithms: Many clusterings with low F-Measure values untruly obtain a high index value.

Prediction Quality. One might argue that a validity index only has to find the best clustering among several candidates—a single outlier that has a very good index value but a poor clustering can completely ruin the applicability of the index. Therefore we measured the F-Measure value that corresponds to the maximum index value for each validity index and each data set. Table 3.4 comprises the results.

	DS1	DS2	DS3	DS4	DS5	DS6	DS7	average
Dunn	0.73	0.73	0.78	0.66	0.83	0.80	0.74	0.75
D.-B.	0.77	0.77	0.65	0.55	0.56	0.64	0.36	0.61
$\overline{\rho}$	0.73	0.84	0.78	0.98	0.83	0.68	0.66	0.79

Table 3.4: The table shows for each index the F-Measure values that belong to its top-rated clusterings. Since the reference categorization $\mathcal{C}^*$ was among the evaluated clusterings, a perfect prediction corresponds to the F-Measure value of 1.

3.4.5 Concluding Remarks

Clustering algorithms are considered as a technology that has the potential to automatically categorize document sets. Different clustering algorithms produce different clusterings, and cluster validity measures must be applied to identify among a set of clusterings the most valuable one. Internal cluster validity measures assess structural properties of a clustering—they hence are in the role of objective measures. The key question in this connection is whether or not an objective measure can be used to capture a user's information need.

In the field of automatic document categorization the information need corresponds to the categorization quality of a clustering $\mathcal{C}$. Given a reference categorization $\mathcal{C}^*$ the categorization quality of $\mathcal{C}$ can be quantified with the achieved precision and recall values. I. e., in the field of document categorization an answer to the above question can be given by analyzing the correlation of cluster validity measures with the F-Measure.

The experiments investigate classical cluster validity measures from Dunn and Davies-Bouldin and present the new graph-based measures Λ and $\overline{\rho}$. As reported in the experiment section, the new $\overline{\rho}$-Measure performed convincingly on both, artificial and genuine clusterings of different document sets, and it outperformed the classical measures in this domain. The Dunn Index performed robust but missed to discover the real interesting artificial clusterings. The Davies-Bouldin index performed well on artificial data sets—however, it was not able to correctly select the best clustering among clusterings that stemmed from a genuine document cluster application.

3.5 The Suffix Tree Document Representation

The last two sections discussed how categorizations for document collections can be generated in an unsupervised way, i.e. they outlined how clustering algorithms work and how validity measures can be used to assess the quality of clusterings. However, both clustering algorithms and validity measures work on abstract representations of the underlying documents and require the statement of a meaningful document similarity or distance function. Regardless of a clustering algorithm's or validity measure's superiority, these methods are only as good as the underlying representations and similarity statements reflect the corresponding semantic relation. In other words, the better two semantically unrelated documents can be separated based on their representation and the associated similarity value, the easier is the clustering job, and the better is the resulting clustering.

In this connection it should be noted that vector-based document models encode no information about the order by which the words occur in a document[8]. A more sophisticated document model that preserves the complete word order information is the suffix tree document model that we introduce here; it defines the similarity between two documents in terms of string overlaps in their common suffix tree.

[8]Some kind of "weak" order information can be introduced by using phrases or just every sequence of n consecutive words, so-called n-grams, instead of single words.

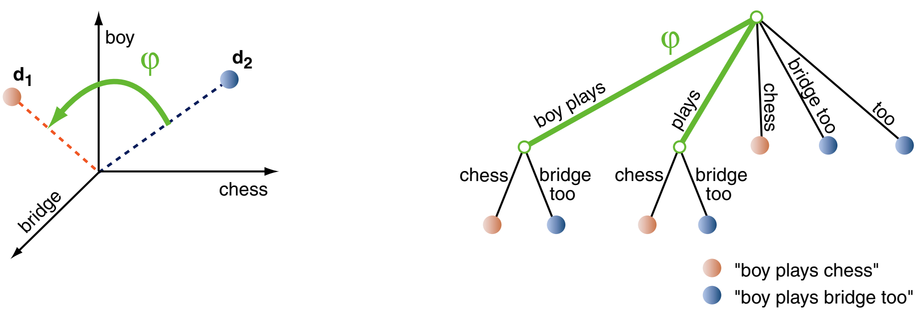


Figure 3.12: Illustration of the two document model types. The left-hand side shows two documents under the vector-based paradigm; the underlying dictionary contains the words "boy", "chess", and "bridge". As similarity function φ the cosine similarity is shown, which corresponds to the cosine of the angle between $\mathbf{d_1}$ and $\mathbf{d_2}$. The right-hand side shows a suffix tree for the documents "boy plays chess" and "boy plays bridge too". Here, the similarity function φ must quantify the portion of the overlap, which corresponds to the green (thick) edges in the graph.

3.5.1 Suffix Trees

The ith suffix of a document $d = w_1 \dots w_m$ is the substring of d that starts with word w_i. A suffix tree of d is a labeled tree that contains each suffix of d along a path whose edges are labeled with the respective words. The construction of a suffix tree is straightforward: The ith suffix of d is inserted by checking whether some edge emanating from the root node is labeled with w_i. If so, this edge is traversed and it is checked whether some edge of the successor node is labeled with w_{i+1}, and so on. If, in some depth k, a node n without a matching edge is reached, a new node is created and linked to node n with an edge labeled with w_{i+k}.

Remarks. Document models and similarity functions φ determine each other: Vector-based document models are amenable to the cosine similarity in first place. The suffix tree document model requires a measure that assesses the similarity between two graphs. Figure 3.12 illustrates both paradigms.

3.5.2 A Closer Look to the Suffix Tree Document Model

In this section we introduce a generic similarity measure for the suffix tree document model. Moreover, we argue that the well-known similarity concepts of the classical document models have their counterpart in the suffix tree document model. Our generic view enables us to seamlessly understand the famous suffix

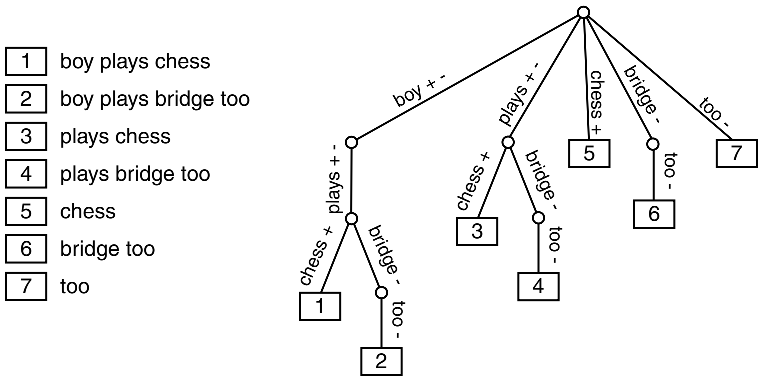


Figure 3.13: A suffix tree for the documents d^+="boy plays chess" and d^-="boy plays bridge too". The seven paths from the root node to the leafs nodes represent the seven suffixes of d^+ and d^-. Edges which are traversed on insertion by suffixes of d^+ are labeled with "+"; likewise, a "-" marks the edges that are traversed by d^-.

tree clustering algorithm of Zamir and Etzioni as a heuristic to efficiently evaluate the graph-based similarity measure for large document collections.

A Graph-Based Similarity Measure. As pointed out above, a document model in the form of a suffix tree preserves full word order information. Here, we introduce a measure that quantifies the similarity of two documents under the suffix tree document model.

Let d^+, d^- designate two documents that are inserted into an initially empty suffix tree T. Each edge e in T gets either labeled "+", "–", or "+–", depending on whether or not e has been traversed while inserting a suffix from d^+ or d^- (cf. Figure 3.13). Moreover, let E denote the edges in T, and let E^+ and E^- denote those edges in E whose label contain a "+" and a "–" respectively. Then the suffix tree similarity φ_{ST} is defined as

$$\varphi_{ST} = \frac{|E^+ \cap E^-|}{|E^+ \cup E^-|}$$

Obviously, φ_{ST} fulfills the following properties of a similarity measure:

(1) *Normalization.* From $0 \leq |E^+ \cap E^-| \leq |E^+ \cup E^-|$ follows that $\frac{|E^+ \cap E^-|}{|E^+ \cup E^-|} \in [0, 1]$.

(2) *Reflexivity.* If $d^+ = d^-$ holds, then $E^+ = E^-$, and consequently $|E^+ \cap E^-| = |E^+ \cup E^-|$ and $\varphi_{ST} = 1$.

(3) *Symmetry.* The symmetry property follows directly from the fact that the insertion order does not affect edge labeling.

φ_{ST} is the Jaccard coefficient of the edge sets E^+ and E^-. Other possibilities to measure the match between the two sets include the Dice coefficient, the cosine coefficient, or the overlap coefficient [111]. Observe that φ_{ST} quantifies the frequencies of suffixes in a Boolean sense, since it is not recorded how often an edge is traversed while inserting suffixes of d^+ and d^-. Put another way, φ_{ST} captures "word order matches" rather than term frequencies.

There are two ways to incorporate term frequencies in φ_{ST}. One possibility is to combine φ_{ST} with a traditional vector space model similarity measure by means of a weighted sum, say, $\varphi_{HYB} = \lambda \cdot \varphi_{ST} + (1-\lambda) \cdot \varphi_{\cos}$, with $\lambda \in [0,1]$. Alternatively, frequency information for each edge can be recorded during the construction of T. The latter approach has the advantage that frequency information for word sequences that are longer than one (suffix frequencies) can be considered for similarity computation.

We construct the suffix tree as described above; all suffixes of d^+ and d^- are inserted into an initially empty tree T. During insertion the functions $n^+(e)$ and $n^-(e)$ are computed, which define for each edge e how often it is traversed when inserting suffixes from d^+ and d^- respectively. Then the similarity value φ_{STF} that incorporates suffix frequencies is given as

$$\varphi_{STF} = \frac{1}{|E|} \sum_{e \in E} \frac{\min\{n^+(e), n^-(e)\}}{\max\{n^+(e), n^-(e)\}}$$

The properties of normalization, reflexivity, and symmetry also hold for φ_{STF}. Note that $n^+(e)$ and $n^-(e)$ capture the term frequencies of d^+ and d^- for all edges e that are incident with T's root.

Research on vector space models has shown that term weighting schemes for document collections that rely on both term frequency and inverse document frequency outperform schemes that are based on only one of these concepts [126]. Note that under the suffix tree document model the inverse document frequency can also be measured for a document collection $D = \{d_1, \ldots, d_n\}$: We construct a suffix tree T for all suffixes of the $d_i \in D$ and associate an initially empty set S with each edge e in T. If a suffix of d_i creates or traverses e, we set $S(e) := S(e) \cup \{i\}$. Since $|S(e)|$ captures the document frequency of the suffix that is represented by the path that starts at the root and ends with e, the inverse document frequency can be measured by $IDF(e) = \log(n/|S(e)|)$, leading to

$$\varphi_{STFIDF} = \frac{1}{|E|} \sum_{e \in E} \frac{\min\{n^+(e), n^-(e)\}}{\max\{n^+(e), n^-(e)\}} \cdot IDF(e)$$

STC: A Fusion Heuristic for the Suffix Tree Document Model. Given a similarity measure like the cosine similarity for the vector space model or one of the suffix tree similarity measures introduced above, the construction of a similarity graph is in $O(n^2)$ for a document collection D of size n. However, [163] introduced the suffix tree clustering algorithm (STC), which runs in $O(n)$ without computing $O(n^2)$ similarity values. In detail, STC is made up of three steps.

Step 1. A suffix tree for all suffixes of each document in $D = \{d_1, \ldots, d_n\}$ is constructed, and each suffix is associated with the set of documents wherein it is contained. In other words, using the notation given above, for each edge e (each of which representing a certain suffix) the set $S(e)$ is computed. The sets $S(e)$ with $|S(e)| \geq 2$ are called "base clusters" and identify the documents d_i with $i \in S(e)$.

Step 2. Each base cluster is assigned a score f, which is a function of $|S(e)|$ and the length of the suffix that is represented by e. In [163] the authors propose f as the product of $|S(e)|$ and the length of the suffix that is represented by e.

Step 3. The k base clusters $S_1, \ldots, S_k$ that score best under f are selected. A similarity graph in which the base clusters form the node set is generated, and an edge between two nodes S_i and S_j is added if the Jaccard coefficient of S_i and S_j is larger than 0.5, say, when $\frac{|S_i \cap S_j|}{|S_i \cup S_j|} > 0.5$. The connected components of this graph form the final clusters.

Analysis of the STC Heuristic. STC has proven to work well on document snippets that are returned by search engines [163], but its properties have not been analyzed yet. As pointed out above STC is a heuristic which is highly efficient, but which also has some drawbacks. The following observations will provide a rationale for some of STC's characteristics.

Non-Exclusiveness. Documents may be associated with several base clusters. Consequently, the documents may appear in more than one of the found categories.

Incompleteness. A clustering that is generated by STC does not necessarily contain all documents of the original collection. An incomplete categorization happens for document collections which comprise documents that share only few short word sequences with the remaining documents. The reason for this behavior is that the emerging base clusters will not score high.

Document frequency based. A base cluster scores higher if the document frequency of its associated suffix increases. This can lead to big clusters, because it is likely that high-scoring base clusters that contain terms with a high document frequency share more than half of their associated documents with other base clusters and consequently are merged in Step 3 of the STC algorithm.

Drifting. Suppose that four base clusters, S_1, S_2, S_3, S_4, are given, where $S_1 = \{1, 2, 3\}$, $S_2 = \{2, 3, 4\}$, $S_3 = \{3, 4, 5\}$, and $S_4 = \{4, 5, 6\}$. Then all documents $d_1, \ldots, d_6$ are merged into a single cluster within Step 3 because the Jaccard coefficient of S_i and S_{i+1} is larger than 0.5. In particular, this single cluster comprises the documents that are associated with S_1 and S_4, which might be completely dissimilar.

Absoluteness. STC considers two documents as similar and assigns them to the same base cluster if they share a rather long suffix or several short suffixes. Regardless of whether their base cluster is merged with other base clusters in Step 3, their base cluster is part of the final clustering. Note that no information about document lengths or suffix mismatches of the rest of those documents is computed, resulting in poor quality clusters. This point is relatively unimportant for short documents—a fact that explains the good performance of STC on document snippets like those returned by search engines.

Topic Generating. Each base cluster is associated with a suffix, which can serve as a label for this cluster. This method solves two basic problems in topic identification for document clusters [134]: word order preservation and topic length determination.

3.5.3 Experimental Evaluation

The purpose of our experiments is twofold. First, we want to gain evidence on STC's character as a heuristic, say, to measure how good the STC algorithm performs on popular document collections compared to clustering algorithms that rely on the new suffix tree similarity measures or the traditional vector space similarity measures. Second, we want to answer the question to which extent word order preservation improves clustering performance. For this purpose we evaluated STC and the clustering algorithms MajorClust [139] and Group Average Link using the discussed similarity measures on several categories drawn from RCV1 [115].

Document Sets. The number of categories in our test data varies from three to six. For each category between 50 and 300 documents were drawn randomly from the entire category. The data sets have different sizes and class numbers, and we investigate uniformly as well as non-uniformly distributed category sizes. Table 3.5 gives an overview of the constructed data sets. The document preprocessing involves parsing, stop word removal according to standard stop word lists, and the application of Porter's stemming algorithm [106].

	DS1	DS2	DS3	DS4	DS5	DS6
# categories	3	4	3	5	4	6
# documents	300	400	500	600	700	800
uniformly distributed	no	yes	no	yes	yes	no

Table 3.5: Overview of the constructed document sets.

Results. We employed STC and the graph-based clustering algorithms MajorClust and Group Average Link to cluster the constructed document sets. For MajorClust and Group Average Link the underlying document models were varied by computing the edge weights according to the cosine similarity measure using the $tf \cdot idf$ term weighting scheme, and the new suffix-tree-based similarity measures. For the hybrid measure, we used $\varphi_{HYB} = \lambda \cdot \varphi_{ST} + (1 - \lambda) \cdot \varphi_{\cos}$ and found that $\lambda = 0.2$ ($\lambda = 0.5$) works well for MajorClust (Group Average Link). Table 3.6 and 3.7 show the achieved F-Measure values [111] for MajorClust and Group Average Link respectively. Note that performance improvements of up to

40% for the new hybrid similarity measure in comparison with the cosine simililarity measure can be observed. The outlined disadvantages of STC are reflected in STC's F-Measure values.

	DS1	DS2	DS3	DS4	DS5	DS6	average
STC	0.55	0.40	0.61	0.33	0.40	0.34	0.44
$\varphi_{\cos}$	0.80	0.60	0.62	0.67	0.66	0.49	0.64
φ_{ST}	0.55	0.46	0.61	0.38	0.45	0.55	0.50
φ_{STF}	0.82	0.70	0.70	0.68	0.76	0.55	0.70
φ_{STFIDF}	0.60	0.60	0.71	0.64	0.78	0.62	0.65
φ_{HYB}	0.84	0.83	0.72	0.74	0.93	0.64	0.78
Improvement in %	5%	38%	16%	10%	40%	31%	22%

Table 3.6: The table shows the achieved F-Measure values for STC (first row), for MajorClust with the traditional similarity measure $\varphi_{\cos}$ on $tf \cdot idf$ vectors (second row), and for MajorClust with the new suffix-tree-based similarity measures (remaining rows). The improvement refers to φ_{HYB} with respect to $\varphi_{\cos}$.

	DS1	DS2	DS3	DS4	DS5	DS6	average
STC	0.55	0.40	0.61	0.33	0.40	0.34	0.44
$\varphi_{\cos}$	0.82	0.63	0.69	0.55	0.78	0.51	0.64
φ_{ST}	0.55	0.40	0.61	0.33	0.40	0.55	0.47
φ_{STF}	0.83	0.64	0.71	0.57	0.85	0.63	0.71
φ_{STFIDF}	0.84	0.72	0.71	0.64	0.80	0.60	0.72
φ_{HYB}	0.84	0.74	0.74	0.66	0.92	0.70	0.77
Improvement in %	2%	18%	7%	20%	18%	37%	17%

Table 3.7: The table shows the achieved F-Measure values for STC (first row), for Group Average Link with the traditional similarity measure $\varphi_{\cos}$ on $tf \cdot idf$ vectors (second row), and for Group Average Link with the new suffix-tree-based similarity measures (remaining rows). The improvement refers to φ_{HYB} with respect to $\varphi_{\cos}$.

3.5.4 Concluding Remarks

Both the classical vector space model and the suffix tree model play an important role in text processing applications: the VSM predominates the information retrieval domain, while suffix trees are employed as data structure for string algorithms. Interestingly, these models are used in an isolated way: there has not been an attempt to combine their different properties for similarity measurement in IR applications.

Our experiments clearly indicate that word order preservation in the document model influences similarity computation to a greater extent. The combination of a vector-space-based similarity measure with a suffix-tree-based similarity measure can lead to a significant improvement of clustering performance, regardless of the chosen clustering algorithm. Moreover, we identified properties of the STC heuristic that explain why this approach can not keep up with any other of the investigated clustering settings on the RCV1.

3.6 Topic Identification

Assume that a categorization $\mathcal{C}$ of a document set D is determined using an unsupervised approach. To present this categorization to a user, it is convenient to label the individual categories with characteristic terms. Amongst others, these terms called category labels[9] should characterize the content of the associated category with respect to the remaining categories. This property implies that it should summarize a category's content and that it should discriminate a category from the other categories. This section resumes desired properties of category labels and reviews algorithms to generate such labels. Moreover, a novel evaluation methodology as well as experimental evaluations of topic identification approaches for flat categorizations are contributed.

3.6.1 Formal Framework

Desired properties for category labels are expressed in the formal framework from [134]; they are resumed in the following. For a categorization $\mathcal{C}$ let $W_d = \{w_{d_1}, \ldots, w_{d_n}\}$ denote the word set for document d, and let $W = \bigcup_{d \in D} W_d$ denote the entire word set underlying D. Term frequency and inverse document frequency of a term w are expressed by the functions $\mathit{tf}(d, w)$ and $\mathit{idf}(w)$, respectively.

Several clustering algorithms define a hierarchy or can be applied in a recursive manner, this way defining a hierarchy $H_{\mathcal{C}}$ on $\mathcal{C}$. $H_{\mathcal{C}}$ is a tree whose nodes

[9]Category labels are also called category names or topic identifiers.

correspond to the categories in $\mathcal{C}$ from which one is marked as root node. Given two categories, $C_i, C_j, C_i \neq C_j$, we write $C_i \succ C_j$ if the corresponding nodes in $H_\mathcal{C}$ lie on a common path emanating at the root and if C_i is closer to the root than C_j.

Topic identification means the construction of a function $\tau : \mathcal{C} \to 2^W$ that assigns to each element $C \in \mathcal{C}$ a set $W_C \subset W$. The following properties are generally desired for a labeling function τ:

(1) *Unique.* $\forall_{\substack{C_i,C_j\in\mathcal{C}\\ C_i\neq C_j}} : \tau(C_i) \cap \tau(C_j) = \emptyset$

(2) *Summarizing.* $\forall_{C\in\mathcal{C}}\ \forall_{d\in C} : \tau(C) \cap W_d \neq \emptyset$

(3) *Expressive.* $\forall_{C\in\mathcal{C}}\ \exists_{w'\in\tau(C)}\ \forall_{d\in C}\ \forall_{\substack{w\in W_d\\ w\notin\tau(C)}} : \mathit{tf}(d,w) \leq \mathit{tf}(d,w')$,

where $\mathit{tf}(d,w)$ designates the term frequency of term w in document d.

(4) *Discriminating.* $\forall_{\substack{C_i,C_j\in\mathcal{C}\\ C_i\neq C_j}}\ \exists w' \in \tau(C_j) : \frac{1}{|C_i|}\mathit{tf}_{C_i}(w') \ll \frac{1}{|C_j|}\mathit{tf}_{C_j}(w')$,

where $\mathit{tf}_C(w)$ is the term frequency of w in category C, say,

$\mathit{tf}_C(w) = \sum_{d\in C} \mathit{tf}(d,w)$.

(5) *Contiguous.* $\forall_{C\in\mathcal{C}}\ \forall_{\substack{w',w''\in\tau(C)\\ w'\neq w''}}\ \forall_{d\in C}\ \exists_{w_i,w_{i+1}\in W_d} : w_i = w' \wedge w_{i+1} = w''$

(6) *Hierarchically Consistent.*

$\forall_{\substack{C_i,C_j\in\mathcal{C}\\ C_i\neq C_j}} : C_i \succ C_j \Rightarrow P(w_i|w_j) = 1 \wedge P(w_j|w_i) < 1$,

where $w_i \in (W_{d_i} \cap \tau(C_i))$, $w_j \in (W_{d_j} \cap \tau(C_j))$, $d_i \in C_i$, $d_j \in C_j$.

(7) *Irredundant.* $\forall_{C\in\mathcal{C}}\ \forall_{\substack{w',w''\in\tau(C)\\ w'\neq w''}} : w'$ and w'' are not synonymous.

The stated properties formalize ideal constraints which, in the real world, can only be approximated. Note that merely Property 6 requires the existence of a category tree $H_\mathcal{C}$[10]. Finally note that hierarchical consistency and irredundancy are practically impossible to achieve if no external knowledge is provided.

[10]A discussion of desired properties for hierarchies can be found in [78].

3.6.2 Related Work

Like the stated properties show, topic identification is related to keyword extraction and text summarization. The main difference for topic identification is that each identified topic refers to a *set* of documents instead of a single document. In the following we resume existing topic identification approaches, which can be classified according to the underlying categorization type: flat or hierarchical.

Topic Identification in Flat Categorizations. In past work on keyword extraction from English texts, several term features have been proposed to identify meaningful keywords; they include the following: (1) first occurrence measured by a term's offset from the beginning of a document, (2) term frequency and, for document collections, inverse document frequency, (3) co-occurrence information [157, 89, 146]. It is unclear how some of these features can be generalized for topic identification; e. g. it is unclear how first occurrence can be defined and if first occurrence is a significant feature with respect to a set of documents. In general, it is questionable if features for keyword identification are suited or can be adapted for topic identification.

Popescul and Ungar propose a labeling algorithm for flat categorizations using frequency and predictiveness information of terms [105, 161]. In particular, the function

$$f_C(w) = P(w \mid C) \cdot \frac{P(w \mid C)}{P(w)}$$

measures the score for word w to be topic identifier in cluster C. Here, $P(w \mid C)$ denotes the conditional probability to draw w from a document in C, and $P(w)$ is the probability to draw w from the whole collection. The first factor in f_C makes sure that more frequent terms within a category are preferred, while the second factor measures the predictiveness[11] of w for C, i. e. the discrimination power. These factors mean to optimize points (2), (3), and (4) from the desired properties. However, point (1) is not addressed by this approach.

Topic Identification in Category Hierarchies. Popescul and Ungar also propose a two-step algorithm to label hierarchical document clusterings as follows [105]. Let the cluster hierarchy be modeled as a tree in which each node represents a cluster, and let the nodes contain their associated documents[12]. In the first

[11]Popescul and Ungar remark that the second factor is similar to a mutual information estimator.

[12]Popescul and Ungar assume that only the leaf nodes hold documents.

step, bag-of-word representations of the documents are propagated bottom-up starting at the leaf nodes: at each inner node, the bag-of-word representations of its descendants are coalesced. In a second step, the tree is recursively traversed starting at the root node. For each word that is associated with the current node, a χ^2 independence test is conducted to gain evidence if the word appears equally likely in all of the child nodes. If this is the case, the word is considered being general for all children: It is kept as label in the current node and it is deleted from all children. Else, if the test rejects the independence hypothesis, the conclusion is that the term is specific for one or more child nodes. In this case, it is deleted from the current node. The χ^2 test puts emphasis on points (3) and (6) in the first place, while the bottom-up as well as top-down propagation strategy addresses points (1) and (2) from the formal framework.

The STC algorithm presented in the previous section has the advantage that cluster labels are generated during the clustering procedure. As mentioned above, these labels are path labels in suffix trees in which a set of documents has been inserted. The nodes that score best in terms of frequent visits and depth within the suffix tree are chosen as cluster labels. In contrast to other methods, this labeling approach preserves word order, contributing to property (5) in the formal framework. However, STC has problems fulfilling properties (6) and (7).

Subsumption analysis approaches, which are based on term distribution analyses of document sets, contribute to address property (6) from the formal framework [122, 80, 82]. The underlying consideration is that some terms occur frequently among all documents within a document set, while other terms only occur in some of the documents within the set. The hypothesis is that if both of the mentioned term kinds co-occur with a certain probability, then the terms may be related in that the more frequent term is more general than the other term. In particular, Sanderson and Croft propose the following definition: Term x subsumes term y if

$$P(x \mid y) \geq 1 - \varepsilon \quad \text{and} \quad P(y \mid x) < P(x \mid y)$$

where $P(x \mid y)$ denotes the conditional probability that term x is drawn when term y has already been drawn from a document [122, 81]. A value of $\varepsilon = 0.2$ has been shown to work well in [122].

3.6.3 The WCC Algorithm for Topic Identification

Like above, let D denote a set of documents, and let $\mathcal{C}$ be a clustering of D. Let $\kappa : W \times \{1, \ldots, |\mathcal{C}|\} \to \mathcal{C}$ be the function with

$$\kappa(w, i) = C \Leftrightarrow C \text{ is at rank } i \text{ in a cluster ranking according to } tf_C(w).$$

E. g. $\kappa(w, 1)$ denotes the cluster in which w appears most frequently and $\kappa(w, |\mathcal{C}|)$ is the cluster in which w appears least frequently.

The algorithm WCC consists of two main parts (cf. Algorithm 1). First, a vector $\mathcal{T}$ is constructed, containing tuples of the form $\langle w,\ tf_{\kappa(w,i)}(w)\rangle$, $i \in \{1, \ldots, k\}$, which are sorted according to descending term frequency. Second, the clusters are cycled in a round-robin manner: Each time a cluster is visited, it is assigned one more word unless it holds l terms. According to the top-down processing of tuples from $\mathcal{T}$, the most frequent words from the cluster centroids are covered.

The labeling τ generated by WCC fulfills the properties (1) and (2) according to the parameters k and l (a value of $k = 1$ ensures the validity of property 1); the cluster-wise Round-Robin-strategy aims at fulfilling property (3). A small k helps fulfilling property (4). Computing the κ-values (including sorting) is in $O(k \cdot |W| \cdot \log(k \cdot |W|))$; assigning the labels is in $O(l \cdot k \cdot |W|)$. Since k and l are typically bounded by a small constant, the overall complexity is in $O(|W| \cdot \log(|W|))$.

3.6.4 Experimental Evaluation

Evaluating the quality of topic identification approaches is a difficult concern since no benchmark collection is available, i. e. there are no document clusterings that have been labeled by humans. In the past, different labelings of the same collection were presented to a couple of people who ranked the labeling quality relatively (cf. e. g. [105, 78]). However, although such studies are useful to get hints concerning relative performance, they are neither reproducible nor they give evidence how good the approaches perform in terms of precision and recall. To overcome these weaknesses, we propose statistics that measure to what extent the desired properties from Section 3.6.1 are satisfied.

Algorithm 1 WCC, Weighted Centroid Covering.

Input: $\mathcal{C}$ clustering
l number of terms per label
k maximum occurrence of the same term in different labels

Output: τ labeling

```
WCC($\mathcal{C}, l, k$)

 (1) $\mathcal{T} = \emptyset$;
     FOREACH $C$ IN $\mathcal{C}$ DO
       $\tau(C) = \emptyset$;

 (2) FOREACH $w$ IN $W$ DO
       FOR $i = 1$ TO $k$
         compute $C = \kappa(w, i)$ from $\mathcal{C}$;
         add tuple $\langle w,\ tf_C(w)\rangle$ to $\mathcal{T}$;
       ENDFOR
     ENDDO

 (3) SORT $\mathcal{T}$ according to descending term frequencies;

 (4) FOR $labelcount = 1$ TO $l$
       $assigned = 0$;
       $j = 1$;
       WHILE $assigned < |\mathcal{C}|$ AND $j \leq |\mathcal{T}|$
         let $t_j = \langle w,\ tf_C(w)\rangle$ be $j$th tuple of $\mathcal{T}$;
         IF $|\tau(C)| < labelcount$ THEN
           $\tau(C) = \tau(C) \cup \{w\}$;
           delete $t_j$ from $\mathcal{T}$;
           $assigned = assigned + 1$;
         ENDIF
         $j = j + 1$;
       ENDWHILE
     ENDFOR

 (5) FOREACH $C$ IN $\mathcal{C}$ DO SORT $\tau(C)$;

 (6) RETURN $\tau$;
```

In particular, the following statistics quantify the development of the properties (1)-(4):

1. *Unique.*

$$f_1(\tau) := 1 - \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{|\ \tau(C_i) \cap \tau(C_j)\ |}{|\ \tau(C_i) \cup \tau(C_j)\ |}$$

   If the term sets of each pair of labels are disjoint, $f_1(\tau)$ takes a value of one; here, k denotes the number of categories in $\mathcal{C}$. The closer f_1 is to zero, the more the terms of distinct labels overlap.

2. *Summarizing.*

$$f_2(\tau) := \frac{1}{k} \sum_{C \in \mathcal{C}} \frac{1}{|C|} \sum_{d \in C} \frac{|\ \tau(C) \cap W_d\ |}{|\ \tau(C)\ |}$$

   The closer $f_2(\tau)$ is to 1, the better the label terms cover the documents of the associated category. A value close to 0 indicates that the category labels appear in only a few documents of a category.

3. *Expressive.*

$$f_3(\tau) := 1 - \frac{1}{k} \sum_{C \in \mathcal{C}} \operatorname*{argmin}_{w' \in \tau(C)} \frac{1}{|C|} \sum_{d \in C} \frac{1}{|W_d|} \sum_{\substack{w \in W_d \\ w \notin \tau(C)}} \frac{tf(d,w)}{tf(d,w')}$$

   If for each cluster there exists an expressive term, $f_3(\tau)$ reaches the maximum 1. Observe that f_3 can take negative values for poorly chosen labels.

4. *Discriminating.*

$$f_4(\tau) := 1 - \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \operatorname*{argmin}_{w' \in \tau(C_j)} \frac{|C_j|}{|C_i|} \frac{tf_{C_i}(w')}{tf_{C_j}(w')}$$

   The closer $f_4(\tau)$ is to 1, the more discriminating is τ. Small values of f_4 indicate weaker discriminative power. Like above, f_4 can be negative.

To evaluate the quality of WCC, Popescul and Ungar's method, and the standard keyword extraction algorithm RSP [146], we downloaded 150 documents from the digital library Citeseer[13], each of which containing author-defined key-

[13]http://citeseer.ist.psu.edu/

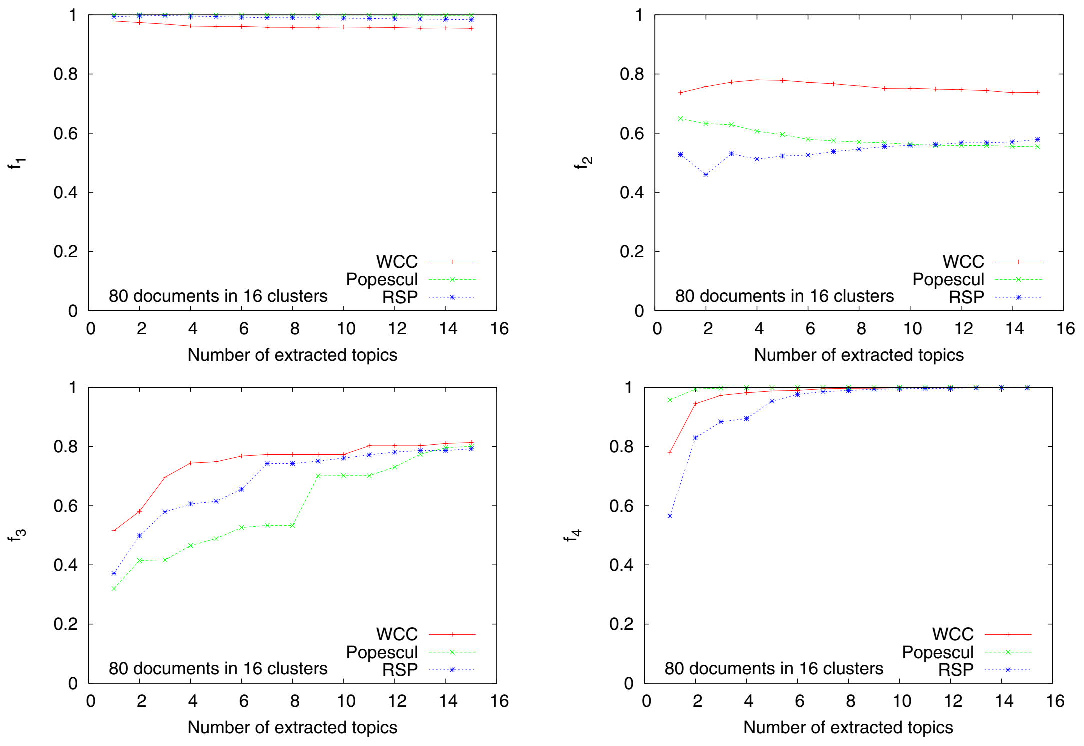


Figure 3.14: The statistics f_1 (top left), f_2 (top right), f_3 (bottom left), and f_4 (bottom right) for the topic identification approaches WCC, Popescul and Ungar's method, and RSP depending on the number of extracted words per label.

words. The evaluation idea is to cluster the documents, label the resulting clustering, compute f_1-f_4, and measure to which extent the identified topic labels appear in the keyword list of at least one document of the associated cluster. To measure the extent, standard precision and recall values can be chosen. Note, however, that the precision value is more important in our scenario since usually only a few words (about one to five) are presented to a user.

The clustering was done on randomly drawn subsets $D' \subseteq D$ with $|D'| = 80$ using k-means, with k varying between 3 and 25. Since the Dunn index has a centroidic view comparable to k-means, we let the Dunn index decide which clustering was best among the generated clusterings. The labelings τ were generated by each of the three mentioned algorithms. For the RSP approach, the documents within a cluster were concatenated to form a single document. The values of the proposed statistics f_1-f_4 are averaged over 10 retries; their values depending on the number of words per label are depicted in Figure 3.14.

Figure 3.15 shows precision and recall curves for the mentioned approaches, again depending on the number of extracted words per label. The values are averaged over 10 retries of the experiment, each time redrawing a new set of documents. The precision curve shows that one to five cluster labels can be identified at very high precision rates.

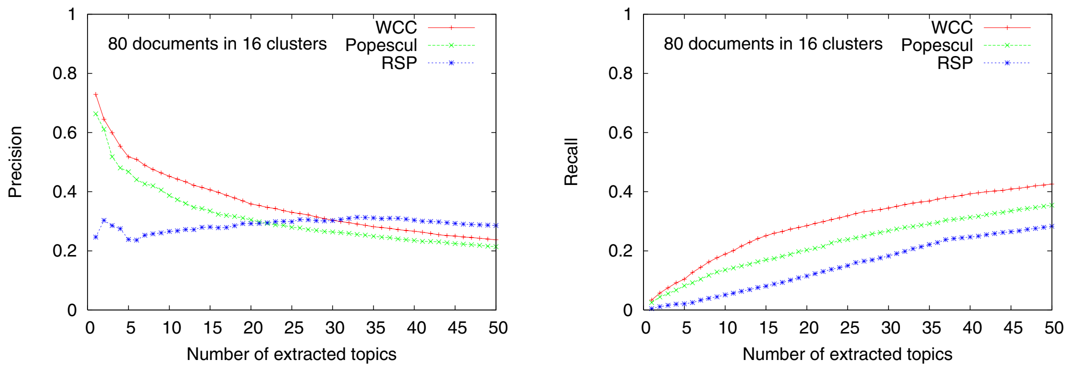


Figure 3.15: Precision (left) and recall (right) of topic identification approaches depending on the number of extracted words per label.

3.6.5 Concluding Remarks

The proposed formal framework of desired properties for category labels shows that topic identification is related to keyword extraction and summarization. The key difference is that topic identification relates to sets of documents—a fact that introduces intra-category as well as inter-category constraints for labels.

In contrast to hierarchical topic identification, not much work has been done in labeling flat categorizations. We proposed the WCC labeling algorithm, which is designed to perform in accordance with the desired properties.

The conducted experiments are ambitious from a combinatoric point of view: only about 15 terms from a cluster's vocabulary of about 2300 distinct terms[14] appear in the keyword list of a document. The results show that especially the properties "summarizing" and "expressive" are very well developed for WCC labelings. The achieved precision values of WCC as well as Popescul and Ungar's approach are surprisingly high.

[14]This number is an average over several generated clusters from our corpus of scientific articles.

3.7 Genre Classification

People who search the World Wide Web usually have a clear conception: They know what they are searching for, and they know of which form or type the search result ideally should be. The former aspect relates to the topic of a document, the latter to the presentation of its content. For example, when searching for the topic “Bayes” in the Internet, a human information miner might be interested in either of the following: a technical article, a biography, or an online book shop. It would be of much help if a search engine could deliver only documents of a desired form, which is here called “genre”. This fact raises two questions:

(1) What are useful genres in the Web, especially when searching the Web, and

(2) How can a detection of these genres be operationalized?

This section will address these questions with the goal to automate genre identification in the context of Web search.

3.7.1 What Does Genre Mean?

As pointed out by Finn and Kushmerick, the term “genre” is used frequently in our culture; e. g., in connection with music, with literature, or with entertainment [39]. Roussinov et al. argue that genre can be defined in terms of purpose or function, in terms of the physical form, or in terms of the document form. And, usually, a genre combines both purpose and form [116].

Here, we are interested in the genre of HTML documents. Several definitions for document genre have been given and discussed in the past [14, 68, 72]. Common to all is that document genre and document content are orthogonal, say, documents that address the same topic can be of a different genre: “The genre describes something about what kind of document it is rather than what the document is about.” [39]. In this way, a genre classification scheme can be oriented at the style of writing, or at the presentation style. When analyzing newspaper articles for example, typical genres include “editorial”, “letter”, “reportage”, “spot news”.

3.7.2 What Does Genre Mean in the WWW?

In the literature on the subject there is more or less agreement on what document genre means and how different genre classes can be characterized. And, at first sight, it seems to be canonical to apply this common understanding to the World Wide Web: Certainly, "advertisement" seems to be a useful genre class, as well as "private homepage". On second sight, however, several difficulties become apparent: Where does a presentation of a company's mission end and where does advertisement begin? Or, does a scientific article on a private homepage belong to the same genre like a photo collection of mom's lovely pet?

Our proposed definition of genre classes for the World Wide Web is governed by two considerations:

- Usability from the standpoint of an information miner, which can be achieved by a what we call "positive" and "negative" filtering. With the former the need for a focused search can be satisfied, while the latter simply extends the idea of spam identification to a diversified genre scheme.

- Feasibility with respect to runtime and classification performance.

The first point means that we want to support people who use the World Wide Web as a huge database to which queries are formulated.[15] The second point states that automatic genre identification shall happen on the fly, in the form of a post-processing of the results of a search engine. This aspect prevents the computation of highly sophisticated features as well as the application of a fine-grained genre scheme.[16] To get an idea which genre classes are considered useful by search engine users, we conducted a user study that is described in detail in Section 3.7.4.

3.7.3 Existing Work

We distinguish the existing work for computer-based genre classification with respect to the underlying corpus, say, whether it is targeted to a particular document collection—like the Brown Corpus, for example—or to the World Wide Web. In the following we outline selected papers.

[15]There are other groups of Internet users who use the Web for amusement, for example.

[16]Crowston and Williams identified about hundred genre classes on the World Wide Web [23].

Corpus-specific genre classification has been investigated among others in [72, 128, 30, 39]. The existing work can further be distinguished with respect to the interesting genre classes and the types of features that have been evaluated. Kessler et al.'s work is based on the Brown Corpus. For the characterization of genre classes they employ so-called genre facets, which are quantified by linguistic and character-level features [72]. Stamatatos et al. use discriminant analysis based on the term frequencies to identify the most discriminative terms with respect to four newspaper genre classes [128]. Dewdney et al. concentrate on different learning approaches: Naive Bayes, C4.5, and support vector machines. They employ about three hundred features including part of speech, closed-class word sets, and stemmed document terms [30]. Rehm proposes a Web genre hierarchy for academic homepages and a classifier that relies on HTML metadata, presentation related tags and unspecified linguistic features. Finn and Kushmerick distinguish between the two genres "objective" and "subjective"; they investigate three types of features sets: the document vector containing the stemmed list of a document's terms without stop-words, features from a part of speech analysis, and easily computable text statistics [39].

Genre classification and navigation related to the World Wide Web is quite new, and only very few papers have been published on this topic. Bretan et al. propose a richer representation of retrieval results in the search interface. Their approach combines content-based clustering and genre-based classification that employs simple part-of-speech information along with substantial text statistics. The features are processed with the C4.5 algorithm; however, the authors give no information about the achieved classification performance [17]. Roussinov et al. present a preliminary study to automatic genre classification: Based on an explorative user study they develop a genre scheme that is in part similar to ours and that comprises five genre groups. However, their work describes an ongoing study, and no recognition algorithm has been implemented [116]. Dimitrova et al. describe how shallow text classification techniques can be used to sort the documents according to genre dimensions. Their work describes an ongoing study, and experience with respect to the classification performance is not reported [33]. Lee and Myaeng define seven genre types for classifying documents from the World Wide Web. Aside from the genre "Q&A" and "Homepage" Lee and Myaeng use also the newspaper-specific genres "Reportage" and

"Editorial". The operationalized feature set is based on a list of about hundred document terms tailored to each genre class [83].

3.7.4 User Study and Genre Selection

Although we have an idea of potentially useful genres, a user study should give insights into the importance of dedicated genre classes. Moreover, it can be used as a basis to select genres for building test collections. As a matter of course, selected genres influence feature selection for automatic classification.

User Study. To get an idea about the expected usefulness of different Web page genre classes were manifold, we decided to inquire a bigger number of search engine users. We developed a questionnaire to shed light on search engine use, usefulness of genre classification, and usefulness of genre classes; in detail, we were interested in the following points.

(1) *Frequency of Search Engine Use.* We expect that experienced search engine users have a clearer idea whether genre classification could be useful or not. We asked the interviewees how often they use search engines. Possible answers were "daily", "once or twice a week", "once or twice a month", and "never".

(2) *Typical Topics for Queries.* As already pointed out, our target audience should use the World Wide Web not only for entertainment, but also as information source. To get an idea what the interviewees search for on the Internet, we let them specify up to 3 typical search topics.

(3) *Usefulness of Genre Classification.* With this question we wanted to figure out if genre filtering is considered as useful in general, i. e. if genre filtering helps to satisfy the user's information need. Possible answers were "very useful", "sometimes useful", "not useful", and "don't know".

(4) *Favored Genre Classes.* We proposed ten genre classes that we found interesting: publications/articles, scholar material, news, shops, link collections, help and FAQ, private portrayals, commercial portrayals, discussion forums, and product presentations. For each of these genres, the interviewees could specify the usefulness in terms of "very useful", "sometimes useful", "not useful", and "don't know".

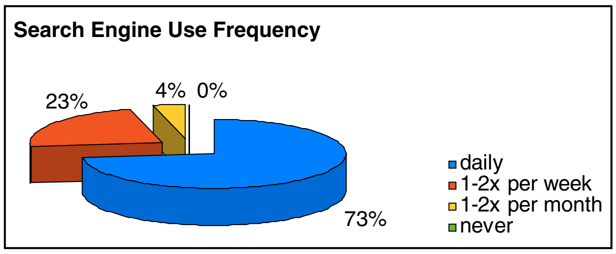


Figure 3.16: Frequency of search engine use. About three quarters of the interrogated students use a search engine on a daily basis.

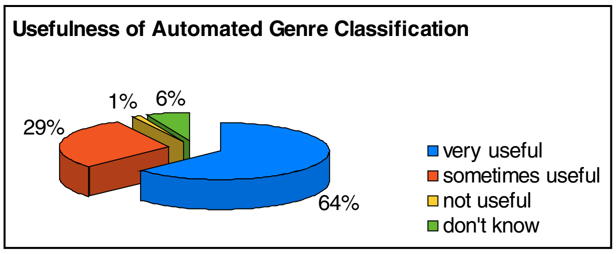


Figure 3.17: Usefulness of genre classification.

(5) *Additional Useful Genre Classes.* We also wanted to find out which additional genre classes could be interesting for the users. Therefore, a set of up to three additional genre classes could be specified and classified into "very useful" and "sometimes useful".

(6) *Comments.* We also gave the interviewees the possibility to comment on the idea of genre classification.

To give the interviewees an idea of genre classification, we gave them a short introduction to genres and their use as positive and negative information filters. As we expect students to frequently use search engines, we asked 286 of them in our university to complete the proposed form. Figure 3.16 shows that we met the right audience: about three quarters of the students use search engines on a daily basis, and nearly the remaining quarter at least once a week.

The most frequently mentioned searches comprise scholar material, shopping and product information, help (discussions and troubleshooting), entertainment (music / games / films / humorous material / news), downloads, health, and programming (in this order). The fact that 64% of the students think that genre classification is very useful, and that another 29% find it sometimes useful shows that there is a strong need to post-process query results (cf. Figure 3.17).

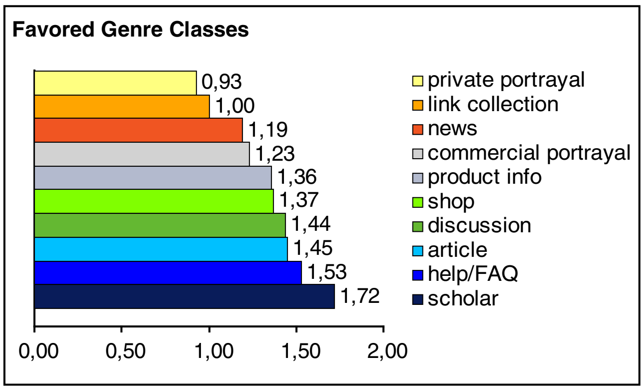


Figure 3.18: The favored genre classes. Higher values indicate a greater expected usefulness.

To make up a ranked list of dedicated genre classes with respect to their usability, we assigned scores on the usefulness of each genre class: “very useful” scored 2 points, “sometimes useful” scored one point, “not useful” scored 0 points. We added the scores for each proposed genre and divided it by the number of interviewees that did not tick “don’t know” on that genre class. The results are depicted in Figure 3.18: scholar material scores best, while private portrayals were not judged as very useful by the interviewees.

Additional genres that were significantly often proposed include Web page spam and download sites. As the given comments and some given specifications of spam let conclude, spam comprises in this context (a) paid links, (b) sites that try to install dialers, and (c) sites that are only used to improve a site’s ranking in search engines. Other propositions included topics (and not genres) like pornography. The comments were encouraging and often asked for operationalization.

Genre Selection. An inherent problem of Web genre classification is that even humans are not able to consistently specify the genre of a given page. Take for example a tutorial on machine learning that could be either classified as scholar material or as article. In general, scholar material can be seen as a super-genre that covers help, article, and discussion pages; therefore scholar material was not chosen as a genre on its own. Another finding is that most product information sites are combined with a shopping interface, which renders a discrimination of the shops and products impossible.

To cut a long story short, we finally ended up with the following eight genre classes:

(1) *Help.* All pages that provide assistance, e. g. Q&A or FAQ pages.

(2) *Article.* Documents with longer passages of text, such as research articles, reviews, technical reports, or book chapters.

(3) *Discussion.* All pages that provide forums, mailing lists, or discussion boards.

(4) *Shop.* All kinds of pages whose main purpose is product information and sale.

(5) *Portrayal (non-priv).* Web appearances of companies, universities, and other public institutions. I. e., home or entry or portal pages, descriptions of organization and mission, annual reports, brochures, contact information, etc.

(6) *Portrayal (priv).* Private self-portrayals, i. e., typical private homepages with informal content.

(7) *Link Collection.* Documents which consist of link lists for the main part.

(8) *Download.* Pages on which freeware, shareware, demo versions of programs etc. can be downloaded.

Although not every document can be rigorously assigned to a single class, our scheme reflects the genre assessment of many human information miners: A scientific article or a link collection, for instance, is still distinguished as such, independently of the domain holder's form of organization where the document is hosted.

3.7.5 Features for Genre Classification

With respect to the investigated features the existing literature on genre classification falls into two groups: Classifiers that rely on a subset of a document's terms (sometimes called bag-of-words, BOW), and classifiers that employ linguistic features along with additional features relating to text statistics. This section gives an overview of these features. In particular, we introduce features that are based on the frequency class of a word as well as concentration features for closed-class word sets. Moreover, we suggest URL analyses with respect to the closed-class word sets.

Word Frequency Class. The frequency class of a word is directly connected to Zipf's law and can be used as an indicator of a word's customariness. Let D be a document collection, and let $tf_D(w) := \sum_{d \in D} tf(d_i)$ denote the frequency of a word w in D, and let $r(w)$ denote the rank of w in a word list $\mathcal{T}$ of D, which is sorted by decreasing frequency.[17]

In accordance with [148] we define the word frequency class $c(w)$ of a word $w \in \mathcal{T}$ as $\lfloor \log_2(f(w^*)/f(w)) \rfloor$, where w^* denotes the most frequently used word, i. e. $w = \arg\max_{w \in \mathcal{T}}(tf_D(w))$. In the Sydney Morning Herald Corpus [28], w^* denotes the word "the", which corresponds to the word frequency class 0; the most uncommonly used words within this corpus have a word frequency class of 19. The intuition to use the word frequency class as feature is the expectation that articles use a more specialized speech than e.g. shops. The complexity of speech is expected to be reflected in the average word class.

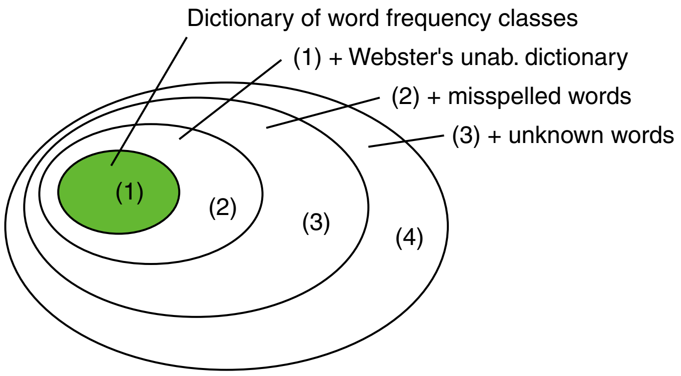

Dictionary of word frequency classes
(1) + Webster's unab. dictionary
(2) + misspelled words
(3) + unknown words
(1)
(2)
(3)
(4)

Figure 3.19: The figure shows the inclusion relation of the used word sets. Note that the sets (3) and (4) are only implicitly defined, by means of the Levenshtein distance and the "not-found" predicate respectively.

Based on the Sydney Morning Herald Corpus, which contains more than 38,000 articles, word frequency classes for about one hundred thousand words have been computed. This dictionary is shown as set (1) in Figure 3.19. The other sets in the figure evolve in a natural manner as supersets of (1): Webster's

[17]Zipf's law states that $r(w) \cdot f(w)$ is constant.

unabridged dictionary (2), the set of misspelled words (3), and the set of unknown words (4). Observe that set (3) comprises all words from the sets (1) and (2) as well as words found in the Levenshtein distance of one [85]. We use these sets to define the following features:

- average word class
- average number of misspelled words
- average number of words not found in Webster's unabridged dictionary

Syntactic Group Analysis. A syntactic group analysis yields linguistic features that relate to several words of a sentence. Such analyses quantify the use of tenses, relative clauses, main clauses, adverbial phrases, simplex noun phrases, etc. Since the identification of these features is computationally expensive, we have omitted them in our analysis. Dewdney et al., however, also include the transition in verb tense within a sentence in their analysis [30].

Part-of-Speech Analysis. Part-of-speech analysis groups the words of a sentence according to their function or word class. Part-of-speech taggers analyze a word's morphology or its membership in a particular set. In this connection one differentiates between so-called open-class word sets and closed-class word sets, where the former do not consist of a finite number; examples are nouns, verbs, adjectives, or adverbs. Examples for closed-class word sets are prepositions and articles. For our analysis we have employed the part-of-speech tagger of the University of Stuttgart [149]. Table 3.9 and 3.10 list the actually used word classes.

Other Closed-Class Word Sets. Aside from word classes that relate to grammatical function, we have also constructed other closed-class word sets that may be specific to a certain genre: currency symbols, help symbols, shop symbols, link symbols, download symbols, homepage symbols, months, days, countries, first names, and surnames. Table 3.8 shows examples for some elements of some of these sets.

In connection with closed-class word sets it is usual that feature values are measured as the proportion of words from an individual closed-class word set with respect to the total word count within a document. However, since characteristic terms for a genre are often concentrated in one place within a document[18], it is reasonable to measure for a document the maximum concentration of terms from these sets within a sliding term window [136]. Moreover, the appearance of these terms in the URL of a Web document is also a good genre hint. Table 3.9 shows for which closed class word sets we also computed concentration and URL containment features.

Table 3.8: Example terms in closed-class word sets.

Word set	Example members
currency	$, EUR, DLR, pound
discussion	forum, subject, post, views, re:,next, thread
download	Windows, Linux, zip, download
help	FAQ, Q&A, support
homepage	name, ˜, address, phone, homepage
shop	buy, now, purchase, add, to, cart

Text Statistics. Under the label "text statistics" we comprise features that relate to the frequency of easily accessible syntactic entities: clauses, paragraphs, delimiters, question marks, exclamation marks, or numerals. Counts for these entities are put in relation to the number of words of a document. Kessler et al. designate features of this type as "character-level cues" [72]; Finn and Kushmerick designate such features as "hand-crafted" [39].

Presentation-Related Features. This type of features relate to the appearance of a document. They include frequency counts as well as particular HTML-specific concepts and stylistic concepts. To the former we count the number of figures, tables, paragraphs, headlines, or captions. The latter comprises statistics related to the usage of colors, hyperlinks (anchor links, site-internal links, Internet links), URL specifications, mail addresses, etc.

[18]Examples include table headlines in discussion forums, in which terms like "subject", "poster", "views" etc. can be found. Another example are download sites, in which version information like operating system names or file extensions from different binary formats can be found.

Table 3.9: Feature set A consists of the listed features. The averages are taken with respect to the total word count within a Web document. For closed word sets, the number of appearances in the URL of a page as well as their maximum concentration within the page were additionally measured.

Feature type	Feature set A
Presentation related	avg. # of <p> tags
	avg. # of <ul> tags
	avg. # of
 tags
	avg. # of anchor links
	avg. # of links same domain
	avg. # of links foreign domain
	avg. # of mail links
	avg. # of <img> tags
	avg. # of <tr> tags
Closed word sets	avg. word frequency class
	conc. / avg. # of currency symbols
	URL / conc. / avg. # of help symbols
	URL / conc. / avg. # of shop symbols
	URL / conc. / avg. # of link symbols
	URL / conc. / avg. # of download symbols
	URL / conc. / avg. # of date symbols
	URL / conc. / avg. # of homepage symbols
	URL / conc. / avg. # of first names
	URL / conc. / avg. # of surnames
	avg. # of words that do not appear in Webster's dictionary
Text statistics	avg. # of question marks
	avg. # of letters
	avg. # of digits
	avg. # of dots
	avg. # of semicolons
	avg. # of colons
	avg. # of commas
	avg. # of exclamation marks

Table 3.10: Feature set B extends feature set A by ten additional features. The averages are taken with respect to the total word count within a Web document.

Feature type	Feature set B
Presentation-related Closed word sets Text statistics	identical to feature set A
Part of speech	avg. # of nouns
	avg. # of verbs
	avg. # of rel. pronouns
	avg. # of prepositions
	avg. # of adverbs
	avg. # of articles
	avg. # of pronouns
	avg. # of modals
	avg. # of adjectives
	avg. # of alphanumeric words

Constructed Feature Sets. As our concern is genre classification of search results, the classification should be done “on the fly”, as a post-processing step. Since a user usually waits actively for search results, the features must be computed quickly. We propose a split of the mentioned features with respect to computational effort as follows.

(1) *Features with Low Computational Effort.* These features comprise text statistics, which can be acquired at parse time by means of counters.

(2) *Features with Medium Computational Effort.* All features that are word-related and that require dictionary lookups or superficial parsing. These are closed-class word sets, word frequency class and presentation related features.

(3) *Features with Higher Computational Effort.* This class comprises features that rely on grammar analyses. Syntactic group analysis features and part-of-speech related features fall in this category.

It should be clear that feature category (1) is not powerful enough to discriminate between the genre classes solely. As a consequence, we built a feature set that comprises features of (1) and (2), and a feature set that makes use of all three feature classes. The Tables 3.9 and 3.10 show the details.

3.7.6 Experimental Evaluation

Since no benchmark corpus is available for our concern, we compiled a new corpus with Web documents and analyzed statistical properties of the two feature sets with respect to them. We employed classifiers in the form of discriminant functions from discriminant analyses to test the achievable classification performance. Moreover, we analyzed the classification performance for genre-specific searches and typical user groups. The following subsections outline our experiments.

Corpus Compilation. The compiled corpus of Web documents is described in Table 3.11. Each element in the corpus represents a single HTML document; documents that are composed of frames and Flash elements were discarded. We then generated two distinct representations of each corpus according to the feature sets (see Table 3.9 and Table 3.10).

Statistical Analyses. We conducted a discriminant analysis (linear model, incremental variable selection according to Wilks Lambda, a-priori probability uniformly distributed) to get an idea of the classification performance of the selected features. Table 3.12 shows a confusion matrix that belongs to feature set B. The results range from acceptable to very good—articles, discussion, and download pages are detected with a very high precision of about 85%, and the remaining genres are detected with a good performance. However, about 80% classification performance for cross-validated data (ten-fold) on a huge corpus with eight classes appears still very good to us.

The scatter plot in Figure 3.20 shows a clear separation of help, link list, and discussion from the remaining genres using only the first two discriminant functions. Like the analysis has shown, the first two discriminant functions capture

Genre	# of Documents
article	181
discussion	242
download	201
help	198
link list	233
portrayal (non-priv)	213
portrayal (priv)	193
shop	246
sum	1707

Table 3.11: Composition of the Web document corpus.

Table 3.12: Ten-fold cross-validated confusion matrix. It shows the percentage of correctly classified documents on the diagonal and summarizes the percentage of misclassified documents with respect to other genres. The average classification performance is about 81%.

	Article	Discussion	Download	Help	Link Collection	Portrayal (non-priv)	Portrayal (priv)	Shop	total
Article	84.0%	0.6%	1.1%	3.3%	0.6%	8.3%	1.1%	1.1%	100.0%
Discussion	3.3%	86.0%	1.7%	1.7%	0.8%	3.7%	1.2%	1.7%	100.0%
Download	1.0%	1.0%	81.1%	2.0%	0.0%	10.0%	1.0%	4.0%	100.0%
Help	9.6%	2.0%	0.5%	78.3%	0.0%	5.6%	2.0%	2.0%	100.0%
Link Collection	1.3%	0.9%	1.7%	0.0%	79.0%	12.0%	2.1%	3.0%	100.0%
Portrayal (non-priv)	6.1%	0.9%	6.1%	0.5%	2.8%	76.5%	2.8%	4.2%	100.0%
Portrayal (priv)	6.7%	0.0%	1.6%	0.0%	1.0%	4.7%	85.5%	0.5%	100.0%
Shop	0.8%	1.2%	2.0%	2.4%	0.8%	13.4%	0.4%	78.9%	100.0 %

about 46% of the variance.

Classification Results. We conducted 10-fold cross-validated classifications on the 1707 feature sets using discriminant analyses. Each experiment was conducted twice, using feature set A and feature set B, respectively. The first row of Table 3.13 comprises the classification results when classifying one genre against all. The second row comprises the multiclass classification results. To make our results reproducible for other researchers, the corpus is available on request[19].

Specialized Classifiers. Aside from general classification performance, we are interested in the question how efficient classifiers for single genre classifications and typical user profiles can be built. Assumed that a user starts several queries on the same topic, an intelligent search assistant could figure out to which predefined user profile a user probably belongs and apply the corresponding classifier. We conducted classification experiments for the following three user profiles.

(1) *Edu.* This profile is of educational nature and comprises articles, link collections, and help sites.

(2) *Geek.* Geeks are mainly interested in downloads, discussions, articles, link collections, and help sites.

[19]Meanwhile, our results from [92] have been confirmed in subsequent work on our corpus; corresponding values can be found in Boese and Howe [16] and in Santini [123].

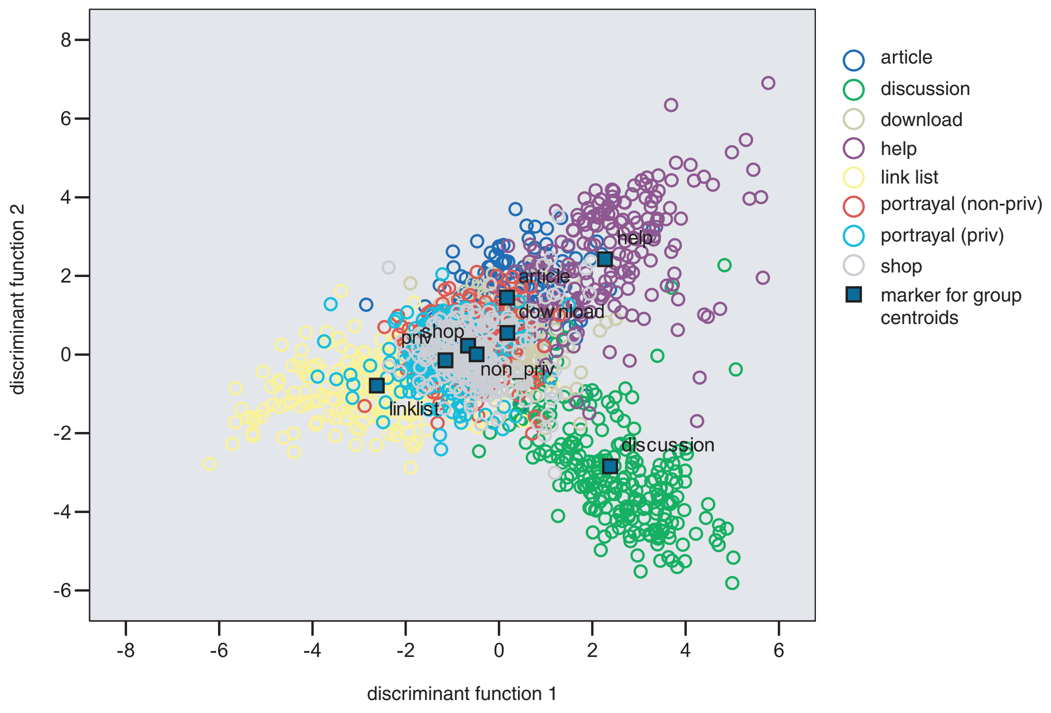


Figure 3.20: The figure shows a scatter plot of the genres according to the first two discriminant functions. The underlying feature set is B.

(3) *Private.* This group comprises individuals that surf the net for shopping and for reading private portrayals.

The performance of the compiled classifiers with respect to both of the feature sets is given in Table 3.13.

3.7.7 Concluding Remarks

We see genre classification as a promising concept to improve the search efficiency and to address the information need of many users that use the World Wide Web as a database. While in the past an automatic detection of genre classes has been demonstrated for newspaper corpora, there is the question whether genre classification can also be applied to the Internet.

A user study has shown the need for advanced page filtering and gave hints on the importance of dedicated Web genre classes. Taken the viewpoint of an Internet information miner we propose the following eight genres: help, article, discussion, shop, portrayals of companies and institutions, private portrayal, link collection, and download. We show that with a small set of features, which cap-

Table 3.13: The table shows the classification performance of specially crafted one-against-all classifiers (first row), multiclass classifiers (second row) and three profile classifiers (remaining rows). The bigger boxes symbolize an aggregation of the genre classes that stand atop of them. Within a box, the upper value shows the classification performance for feature set A, while the lower value shows the performance for feature set B.

	Shop	Portrayal (priv)	Portrayal (non-priv)	Download	Discussion	Article	Link Collection	Help
One against all	94.7% 94.7%	95.9% 96.4%	85.6% 86.2%	95.0% 95.1%	96.0% 96.2%	91.6% 92.0%	96.1% 96.6%	95.5% 96.0%
Multiclass Classification	78.0% 78.9%	85.0% 85.5%	74.6% 76.5%	81.1% 81.1%	86.0% 86.0%	81.8% 84.0%	76.8% 79.0%	77.3% 78.3%
Profile “Edu”	91.8% 91.7%					82.9% 84.5%	78.1% 83.3%	77.3% 79.3%
Profile “Geek”	85.1% 85.3%			82.1% 85.6%	85.1% 85.1%	83.4% 84.5%	77.7% 80.3%	78.3% 78.3%
Profile “Private”	82.5% 82.9%	87.6% 91.2%	92.9% 93.1%					

tures linguistic and presentation-related aspects, text statistics, word set concentration measures, URL pervasion, and word frequency classes, acceptable classification results can be achieved: Our analysis reveals that about 80% of the documents are assigned correctly with a multiclass classifier.

Chapter 4

Tira: A Software Architecture for Personal Information Retrieval

Information retrieval is not a universal answer to a generic information need problem but a collective term for myriad solutions to individual information need problems. To become an effective means, retrieval technology must be adapted to personal information needs, which pertains among others to the following points:

(1) *Personal Data.* Document sources on which retrieval tasks are carried out include local hard drives, the Web, or intranets.

(2) *Personal Preferences.* Typical preferences are language and local settings, or an individual style for result preparation.

(3) *Personal Skills.* This characteristic comprises a user's creativity to formulate queries, his/her ability to improve queries iteratively upon search engine feedback, or background knowledge about the retrieval strategies of search engines.

(4) *Personal Knowledge.* Even when a personal query formulation skill is highly developed, retrieval success still depends on a user's knowledge of the query domain and the underlying collection (e.g. technical terms). This observation applies especially to closed collections or topic-centered collections in corporate intranets.

(5) *Personal IR Tasks.* Advanced personal information needs cannot be suitably addressed with a keyword query approach but require the statement of a tailored IR process. Examples include plagiarism detection, opinion extraction, and filtering according to document quality.

We argue that the current generation of IR tools is not flexible enough to address the above points, especially Point 5. The "course of action" in current IR tools is hard-wired, i. e., a user is restricted to specify a query along with a few parameters and cannot adapt or even design the retrieval process itself. We propose an IR software architecture that follows a service composition paradigm: Given an advanced IR problem, a tailored tool that solves this problem shall be constructed by simply selecting and connecting services from a set of "IR building blocks". Our architecture allows to specify and to store personal IR tasks on a user's personal device and to execute these tasks in a distributed environment.

The remainder of this chapter is organized as follows. Section 4.1 relates IR theory to IR software and motivates the service-oriented approach, Section 4.2 discusses formalisms to specify IR processes, and Section 4.3 introduces architectural concepts behind Tira.

4.1 From IR Theory to IR Software

The operationalization of an IR process is far off from being a standard software engineering task. Current implementation practice is to maintain software libraries that provide generic IR functionality, and to reuse them in other projects. Although this practice has approved in general settings, the IR process design situation comes with properties that allow for a more powerful modeling perspective:

- IR processes are composed of rather autonomous software building blocks, which are called modules here. Basically, each module provides a service that transforms an input data structure into an output data structure. Examples for such modules include import filters, clustering algorithms, validity measures, ranking functions, classifiers, language taggers, and visualization algorithms.

- Information retrieval theory yielded different solutions for one and the same task or for a class of related tasks.[1] Examples include the different approaches to stemming (statistical algorithms, rule-based algorithms [106, 140]) and to keyword extraction (internal versus external methods, corpus-based methods).

- Several tasks within an IR process are addressed with a parameterizable base algorithm.[2] Examples include the language-specific stemming and stopword filtering [107], which take language-specific rules or word lists as their input.

- IR processes are subject to frequent change: they are optimized, tested with new ideas, and adapted to changing information needs.

- Typically, various parts of an IR process can be executed in parallel, especially when documents are analyzed with respect to different objectives. An example is the intrinsic similarity analysis of a document collection with respect to topic, to genre, as well as to writing style [61, 128, 94, 135].

- A set of standard modules, which are useful for virtually any IR process, can be identified. Examples include modules for stemming, modules for stopword removal, and conversion modules for binary formats like Adobe Acrobat (PDF) or Microsoft Word.

The outlined points exhibit the modular nature of IR processes and, in particular, the benefits when this nature is actually exploited. For this we propose a two-step procedure: In a first step an IR process is specified in a diagrammed form; in a second step, this specification is automatically instantiated and deployed as a distributed software system.

4.2 Specification of IR Processes

Consider as an example an IR task where a document shall be categorized according to both a given topic taxonomy and a given genre taxonomy [92]. Figure 4.1

[1]Observe the connection to the Strategy Design Pattern described in [48].

[2]Observe the connections to the Factory Design Pattern and the Decorator Pattern described in [48].

```
Input:   URL u, dictionary dict, stopword list stl.
Output:  genre and topic class for the document at URL u.

Text ht=download(u);
Text plainText=removeHTMLTags(ht);
Text filteredText=removeStopwords(plainText, stl);
FeatureVector topicModel=buildTopicModel(filteredText, dict);

Language lang=detectLanguage(plainText);
FeatureVector presentationFeatures=buildPresentationFeatures(ht);
FeatureVector posFeatures=buildPOSFeatures(plainText, language);
FeatureVector genreModel=union(presentationFeatures, posFeatures);

int topicClass=classifyTopic(topicModel);
int genreClass=classifyGenre(genreModel);

return(topicClass, genreClass);
```

Figure 4.1: IR process for the sample categorization task, specified in pseudo code.

depicts a specification of the underlying IR process in pseudo code: The topic model and the genre model that are constructed from the document found at an URL u form the input for previously built classifiers. Note that several text representations, including HTML text, plain text, and filtered text, are necessary to perform the outlined task.

This kind of specification is current practice in a—what we call—*library-based* modeling approach, but it does not take the nature of IR processes into account: (1) the replacement of a module entails tedious and error-prone code and data structure replacements, (2) it requires in-depth knowledge concerning the library, (3) the exploitation of the concurrency between particular subtasks leads to an inflexible design since such behavior must be hard-wired in the underlying execution model (in the form of threads or remote function calls), (4) the deployment strategy must be hard-wired as well.

We propagate to specify an IR process at a conceptual level, by means of a diagrammed modeling language. In the past, different modeling tools have been proposed for similar purposes; they can be classified according to the following scheme [143]:

(1) control flow dominant or state-oriented: finite state machines, UML state charts

(2) data flow dominant or activity-oriented: data flow graphs, Petri nets, marked graphs, UML activity diagrams

(3) structure-oriented: component connectivity diagrams, UML class diagrams, UML deployment diagrams

(4) time-oriented: UML time diagrams

(5) data-oriented: entity relationship diagrams

(6) hybrid, a combination of the above mentioned principles: control/data flow graphs

Most IR processes can be considered as data flow dominant, i. e., they are invoked by a user who asks to process a query, and none of the involved modules can be executed until its preceding modules have delivered their data.

In addition to prescribing data dependencies, a modeling approach for IR processes must allow for defining concurrency (branching and synchronization) since parts of an IR process may be executed in parallel. Moreover, a modeling approach should support explicit typing in order to analyze module composition constraints with respect to input and output parameters. Finally, depending on the modeling granularity, it can be useful to define iterations on parts of an IR process as well as conditions on the produced data.

In the following, selected modeling tools are discussed with respect to the specification of IR processes.

4.2.1 Petri Nets

A Petri net [102, 143] is a tuple $N = \langle P, T, F, c, w, m_0 \rangle$, where

- $P = \{p_1, \ldots, p_m\}$ is a set of places,
- $T = \{t_1, \ldots, t_n\}$ is a set of transitions,
- $P \cap T = \emptyset$,
- $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation; the elements in F are also called edges,
- $c : P \to \mathbf{N} \cup \{\infty\}$ is a function that defines the capacity restriction of the places,
- $w : F \to \mathbf{N}$ is a function that defines the edge weights,
- $m_0 : P \to \mathbf{N}_0$ is an initial marking with $\forall p \in P : m_0(p) \leq c(p)$.

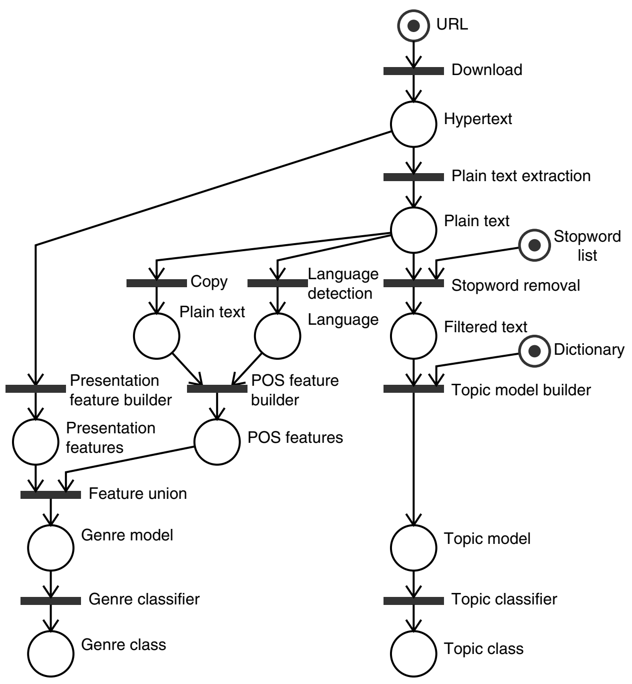


Figure 4.2: IR process for the sample categorization task, specified as Petri net.

A Petri net is a bipartite graph where each $p \in P$ contains up to $c(p)$ tokens. Places are usually displayed as circles; tokens are depicted as bullets within the respective circles. A transition is represented as a bar, and each directed edge from F is displayed as an arrow between the transitions and the places. The marking m_0 defines an initial distribution of tokens, which will change depending on the firing of transitions; $m : P \to \mathbf{N}_0$ denotes an arbitrary marking. Figure 4.2 shows a Petri net of the IR process for our sample categorization task.

Petri Net Semantics For $x \in P \cup T$ let $\bullet x = \{y \mid (y, x) \in F\}$ denote the set of direct predecessors of x; likewise, let $x\bullet = \{y \mid (x, y) \in F\}$ denote the set of x's direct successors. A transition $t \in T$ is called *enabled* under a given marking m if

(1) $\forall p \in \bullet t \setminus t\bullet : m(p) \geq w(p, t)$

(2) $\forall p \in t \bullet \setminus \bullet\, t : m(p) \leq c(p) - w(t, p)$

(3) $\forall p \in t \bullet \cap \bullet\, t : m(p) \leq c(p) - w(t, p) + w(p, t)$

An enabled transition t can fire: for each input place $p \in \bullet t$ a number of $w(p, t)$ tokens is consumed (deleted), and $w(t, p)$ marks are produced (added) for each $p \in t\bullet$.

Discussion. In our scenario the transitions correspond to modules and the tokens represent the data passed between the modules. Petri nets can model sequential as well as concurrent processing, see Figure 4.2: analysis tasks run in parallel, they may be synchronized when displaying results to a user.

Petri nets are well researched: various tools for their analysis and simulation have been developed in the last forty years, including algorithms that determine reachability or deadlocks in a net. However, Petri net tokens are indistinguishable, and hence the handling of different data types cannot be modeled—a fact, which renders Petri nets unusable for our purposes. Another shortcoming relates to the missing possibility to specify a processing order for the produced data: it is questionable whether a FIFO strategy is always desirable.

To circumvent the data type restriction colored Petri nets could be employed [65]. But even with this extension the modeling of control flows such as iterations remains fairly restricted. Since a Petri net cannot "look inside" a token, control flows that depend on the *content* of data (typical for many IR processes) cannot be modeled.

4.2.2 Data Flow Graphs and Control/Data Flow Graphs

A data flow graph $G = \langle V, E \rangle$ is a directed graph where each node represents a task and where each directed edge defines a data flow between its incident nodes. Processing semantics: a task $v \in V$ can only be executed if all $u \in V$ with $(u, v) \in E$ have already been executed. Figure 4.3 shows a data flow graph of the IR process for our sample categorization task.

If Petri net transitions and Petri net places are substituted for a data flow graph's nodes and edges respectively, one obtains a Petri net isomorphic to Figure 4.2. Hence, the shortcomings of Petri nets still apply to data flow graphs. However, the problem that control structures like iterations cannot be modeled is tackled with the more powerful control/data flow graphs, CDFGs. This hybrid approach complements data flow graphs with control flow edges, which can be used to model control flow alternatives as well as iterations. Typically, control flow edges define alternative paths of which exactly one is followed according to a condition.

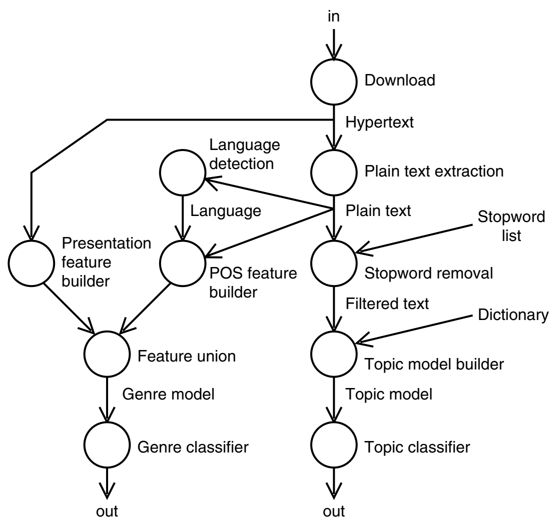


Figure 4.3: IR process for the sample categorization task, specified as data flow graph.

Discussion. CDFGs are powerful enough to model complex IR processes that include branching and iterations. However, the data flow component in Figure 4.3 shows that data types as well as synchronization are only implicitly modeled, by means of labeled edges. This deficit is addressed within the following, more intuitive UML modeling approach.

4.2.3 UML Activity Diagrams

The UML activity diagrams [100] combine novel ideas from Web service flow languages like BPEL [7] with traditional concepts like the token concept from Petri nets in order to specify control flow and data flow between so-called actions. In particular, action nodes, object nodes, and control nodes are connected with directed edges that specify either a data flow or a control flow [58]. Figure 4.4 shows an activity diagram of the IR process for our sample categorization task.

Similar to CDFGs, actions nodes represent tasks, which are software modules in our modeling scenario. Object nodes may be placed between action nodes, symbolizing data objects that are transferred between action nodes. Alternatively, connectors, called "pins", which are attached to the action nodes, can specify the data type that is accepted as input or produced as output by an action node.

Control nodes further divide into decision nodes, merge nodes, fork nodes, and join nodes. Decision nodes delegate control flow exclusively to one of several

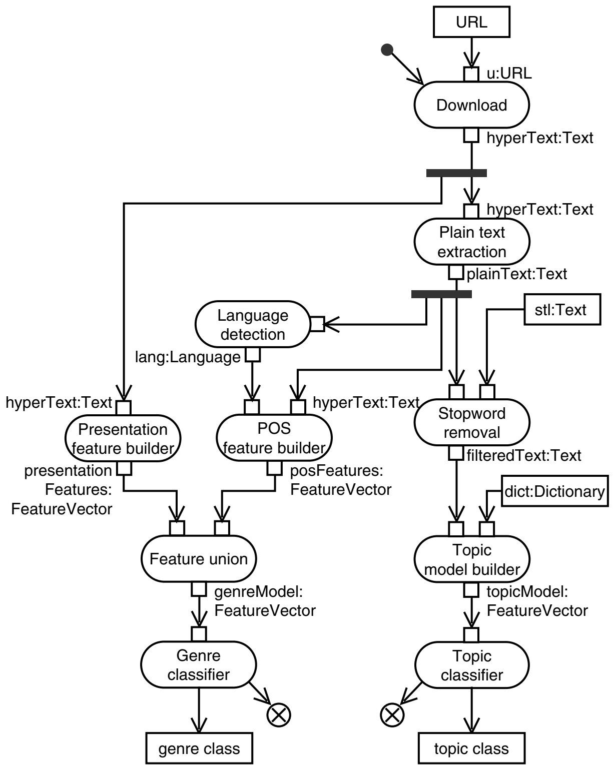


Figure 4.4: IR process for the sample categorization task, specified as UML activity diagram.

possible branches, depending on a condition that is bound to the node; their counterpart are the merge nodes. Concurrency is modeled with fork nodes and join nodes; they indicate the concurrent execution and the subsequent synchronization of control flows and data flows. Finally, buffer nodes, which are specialized object nodes, can be used to define a buffering strategy for concurrent processing.

An activity diagram may be partitioned into so-called “swimlanes” in order to group nodes and edges according to common properties. Such logical groups are oriented at the user-defined semantics (a closed sub-retrieval-task for example) and allow for the structuring of complex IR processes.

Discussion. Apart from being intuitive, UML activity diagrams are widely accepted as modeling tool. Moreover, advanced concepts that allow for the modeling of data streams, parameter sets, stereotypes, action and time events, exceptions, and exception handlers render this diagram form ideal for our purposes. In the upcoming UML 2.1 specification, conditional nodes and iteration nodes, which remind of block diagram elements, will probably be included, making UML activity diagrams even more intuitive for modeling control flows.

4.3 Operationalizing IR Processes with Tira

In terms of the model driven architecture (MDA) paradigm, an IR process specified with a UML activity diagram can be considered as a platform independent model (PIM) [98]: UML activity diagrams are not bound to programming languages, operating systems, middleware, or system architectures. Consequently, to make an IR process operable, a target platform has to be chosen, and the PIM must be transformed into an executable platform specific model (PSM).

In this context a platform denotes the next (lower) abstraction layer on which a particular model is represented in a more concrete form. For example, J2EE and CORBA are possible platforms for a business process implementation, and, the Java Development Kit in turn is a possible platform for a CORBA implementation. As the latter example shows, the transformations along descending platform layers prescribe the path by which a PIM is rendered executable. The OMG denotes a platform that is in-between the PIM and executable code as middleware platform [98].

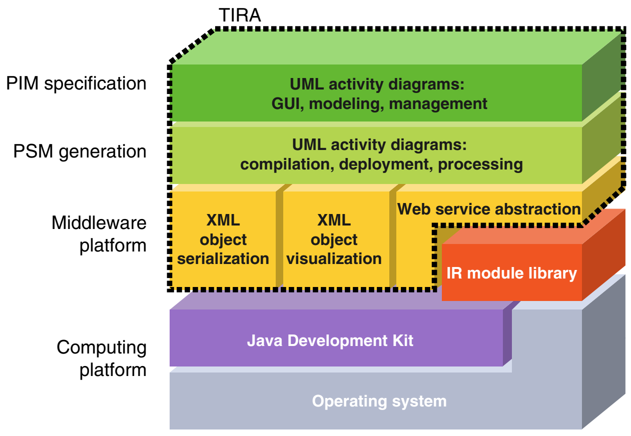


Figure 4.5: The layer architecture of Tira on top of a computing platform. The IR module library is not part of Tira but provides an open and extensible container for IR-related algorithms and data structures.

Just as in other MDA-based application scenarios our objective is to define, to develop, and to implement the transformation of the PIM towards a lower platform layer. However, in contrast to many MDA-based application scenarios we are not interested in the handling of a *variety* of middleware platforms and their related transformations, but in the development of a particular middleware platform that is suited to execute personal information retrieval tasks. Put another way: Our focus is on rapid prototyping, reduced turn-around times, and minimized effort for implementation and test.

Figure 4.5 shows our implemented proposal for the layer architecture of Tira: The input PIM is an IR process, modeled as UML activity diagram; a PIM can be compiled and deployed, becoming an executable PSM this way. Modules with the core IR functionality are comprised in an open IR library; the library encapsulates the modules as Web services to make them transparently usable from a PSM via remote function calls (see also [95]). Data objects that are required or produced by the IR modules are materialized as XML objects.

4.3.1 From PIM to PSM

An activity diagram, either loaded from file or modeled interactively with the Tira GUI, is represented as an object structure in computer memory. The structure is oriented at the UML meta-model [99] and reflects the important elements of activity diagrams, i. e., there are instances of action nodes, fork nodes, etc., which are interconnected by data nodes. The action nodes are bound to IR modules, which in turn are encapsulated as Web services; the data nodes are bound to XML objects.

Figure 4.6 shows the part of Tira's class design that models how action nodes are simulated. Instead of modeling an IR module as subclass of the action node class, action nodes are instantiated using a factory class and configured with a symbolic action name. The factory class looks up the URL of the Web service that is associated with the action name at Tira's service registry and provides the action node instance with this URL. Each IR module that is registered in the service registry comes with a self-description in XML format: input as well as output data types are specified in XML schema. This approach keeps Tira open, since it allows for registering and executing new IR modules without recompiling Tira's source.

The object structure is provided with a Petri-net-like token semantics. Simulating the activity diagram means to check the availability of an action node's input data, to allocate processing resources, and to call the corresponding Web service. On delivery of a Web service result, the associated data tokens are propagated in the object structure.

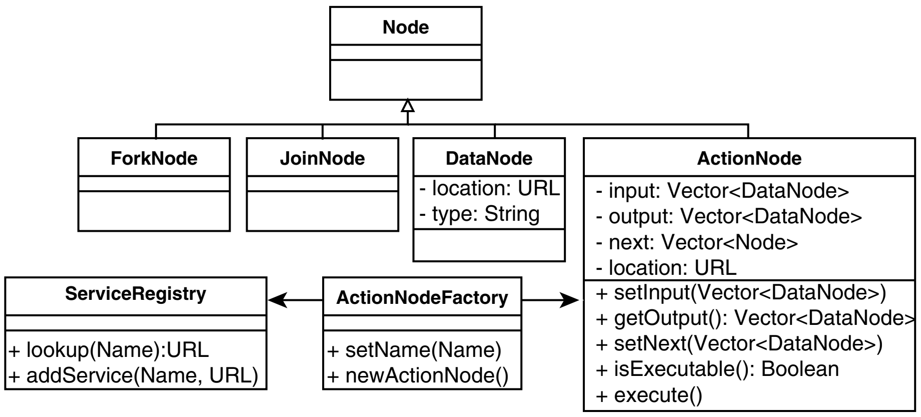


Figure 4.6: A part of TIRA's software design given as UML class diagram.

4.3.2 The TIRA Middleware Platform

The functions in the IR module library take objects as input and return new objects. Instead of supplying the Web service stubs with serializations of these objects, parameter passing is realized with the call-by-name paradigm in its most generic form: a parameter must be a URL, pointing to a serialized XML representation of the respective object. This approach comes with the following advantages.

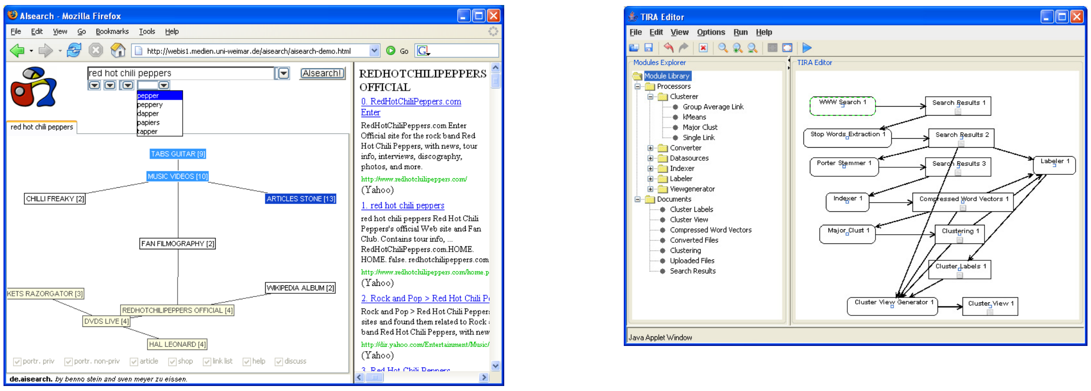
(1) When executing an IR process, a client needs not to transfer the intermediate data between two Web service calls; instead, an invoked Web service fetches the data directly from the given URLs, resulting in reduced data transfer costs.

(2) When two consecutive modules are executed whose corresponding Web services are located on the same server machine, data transfer costs are even lower since the XML files can be directly accessed.

(3) The transfer of URL references instead of data objects enables low-bandwidth machines to be fully functional clients. In particular, home users are able to execute personalized IR processes.

(4) The use of URLs opens the entire World Wide Web as address space for data hosting and data sharing.

Because of the broad acceptance of XML the serialization of data objects as XML streams is the means of choice for data exchange. Within TIRA powerful



REDHOTCHILIPEPPERS OFFICIAL

0. RedHotChiliPeppers.com Enter

RedHotChiliPeppers.com Enter Official site for the rock band Red Hot Chili Peppers, with news, tour info, interviews, discography, photos, and more.
http://www.redhotchilipeppers.com/
(Yahoo)

1. red hot chili peppers

red hot chili peppers Red Hot Chili Peppers's official Web site and Fan Club. Contains tour info, ... RedHotChiliPeppers.com.HOME. HOME. false. redhotchilipeppers.com
http://www.redhotchilipeppers.com/home.p
(Yahoo)

2. Rock and Pop > Red Hot Chili P

Rock and Pop > Red Hot Chili Pepper sites and found them related to Rock band Red Hot Chili Peppers, with new
http://dir.yahoo.com/Entertainment/Music/
(Yahoo)

3. Red Hot Chili Peppers

Figure 4.7: The left screenshot shows our meta search engine, AIsearch, whose underlying IR process is defined in TIRA. The right screenshot shows the TIRA editor for the specification of activity diagrams.

parser generation tools and an XML language binding (JAXB) are responsible for the reading and writing of XML object streams [142].

Intermediate data that is produced during the execution of an IR process is made available for visual inspection, which helps debugging IR processes and lets a user reason about the underlying process. For this purpose the XSL transformation technology is intensively used in TIRA; XSL stylesheets are easy to adapt and to maintain, and they are suited to produce any desired format from the data.

4.3.3 TIRA at Work

Figure 4.7 (left) shows a screenshot of our meta search engine AIsearch, rebuilt as a TIRA application. AIsearch takes as input a keyword query, meta searches commercial Web search engines, and categorizes the found documents according to the identified topics [91]. The underlying IR process extracts the delivered snippets, removes stop words, and stems the remaining words before compressed term vectors are built. Based on the term vectors, a clustering with the MajorClust algorithm [139] is generated, and, for browsing purposes, meaningful text labels for the clusters are constructed with with a statistical text covering algorithm.

Figure 4.7 (right) shows a screenshot of the TIRA activity diagram editor, which is implemented as a Java applet. The left hand side of the applet shows a selection of available IR modules. A double click instantiates a module, which is then displayed graphically on the right hand side of the Applet. The arrows between the modules display the data flow. A click on a data node invokes the associated XSL transformation, which translates the corresponding data to XHTML and displays the results in a browser.

4.4 Concluding Remarks

IR processes have become ubiquitous, be it in the form of search engines at home or at work, on mobile devices or on workstations, or as retrieval components in file systems, document repositories, databases, or knowledge management tools. The reason for this pervasiveness is the growing information need, the diversity of IR tasks, and the desired degree of personalization. Although specialized retrieval algorithms have been developed in the past, less work has been done on modeling and operationalizing IR processes from a software engineering point of view.

This chapter contributes to this aspect. Starting from a discussion of modeling approaches for IR processes we introduced Tira, a flexible MDA solution for the rapid prototyping of tailored IR tools. Tira allows a user to model an IR process as UML activity diagram, which can be transformed to a problem specific model and, based on the Tira middleware platform, executed by the press of a button.

Currently, Tira is a research prototype and further extended and enhanced in our working group. The Tira approach is rather independent of particular IR algorithms and data structures; it is intended to be built around an existing IR module library.

Chapter 5

Conclusion and Outlook

Today, the prevailing form in which search results are presented when querying document collections is as a list of document snippets. Analyses of human search behaviour have shown that most keyword queries are short queries—a fact that leads to long result lists that comprise mainly irrelevant hits. Since human capabilities in information processing are rather limited, this form of search result organization and presentation is questionable. A technique which contributes to solve this problem is document categorization.

Categorization according to topic.

The performance of unsupervised document categorization algorithms is strongly influenced by the underlying document representation. We introduced the suffix tree document model along with new similarity measures, which compile full term order information into the similarity values. Experiments have shown that this model is able to substantially improve the unsupervised categorization performance.

Clustering algorithms require parameters such as the desired number of clusters, density thresholds, and neighborhood specifications. The choice of these parameters affects the clustering quality to a great extent. We introduced $\overline{\rho}$, an internal cluster quality measure, which allows us to compare the quality of clusterings in an unsupervised manner. In particular, it finds the best categorizations in a set of clusterings, each of which has been generated by trying different clustering algorithms or parameters. The experiments have shown that $\overline{\rho}$ correlates better with the F-Measure with respect to reference categorizations of test collections than traditional validity indices.

A categorization has to be presented to a user, i. e. the category structure must be visualized, and topic labels for the found categories must be identified. We formalize desirable properties for category labels and propose the Weighted Centroid Covering algorithm for topic identification. The results of our ambitious experiments are convincing: precision rates between 50% and 75% for up to five extracted topic labels have been achieved. Our new evaluation methodology renders performance measurement reproducible and comparable.

Categorization according to genre.

Document categorization according to genre is considered useful by the vast majority of search engine users; in particular, over 90% of 286 interrogated students considers genre categorization in connection with Web search very useful or sometimes useful.

We identified strong features that allow for building expressive document representations for genre classification. Feasibility studies have shown that a multiclass classification with eight classes using a discriminant analysis can be done at a convincing average performance of about 80% classified correctly. If a single genre category shall be distinguished from the remainder, the classification performance is around 95%. Personalized classifiers achieve error rates in the range between the two mentioned figures.

Software engineering for personalized IR.

Although IR processes exhibit a modular nature, little work has been done in the intersecting fields of software engineering and information retrieval. We propose TIRA, a model driven architecture (MDA) approach for IR process specification and operationalization. This technology separates IR process specification from implementation, and it allows to rapidly develop new prototypes, to quickly adapt IR processes to personal preferences, and to test new ideas at the push of a button.

What comes next?

The first commercial meta search engine that clustered search results was Vivísimo [1]. Although there are other products that provide categorization functionality[2],

[1] `http://www.vivisimo.com`
[2] e. g. `http://www.turbo10.com`

Vivísimo convinces with robust and mature clustering technology. However, like the suffix tree document model, the new validity inices, and the analysis of the STC heuristic has shown, there is still room for optimizing categorization performance.

Meanwhile, two-class genre search engines have evolved; they include Froogle[3], a search engine for shops, and Google Scholar[4], an article search engine. However, there is still no Web search engine that lets a user specify which genres to include or exclude from query results. We expect that a search engine providing such a functionality would be widely appreciated. However, to put such an engine to work, it is advantageous to classify documents according to genre at indexing time since an on-the-fly computation of the according document representation may be too expensive in terms of runtime.

Tira performs well in our labs; however, it must stand the industry test, and the modeling granularity must prove to be practical in a wider range of application scenarios. Also, a deployment of the PIM that is given in the form of a UML activity diagram to high-performance platforms would be interesting to investigate; e. g. theoretical models for the automatic deployment of IR process components to architectures like P2P nets, cluster computers, or grids would be interesting. In particular, the associated tasks to estimate job lengths opens a new field in IR; likewise, performance guarantees would be highly appreciated, especially in distributed commercial applications.

[3]`http://www.froogle.com`
[4]`http://scholar.google.com`

Bibliography

[1] Dimitris Achlioptas and Frank McSherry. On spectral learning of mixtures of distributions. In *COLT*, pages 458–469, 2005.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.

[3] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington D. C., May 1993. ACM Press.

[4] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994*, 1994.

[5] Giambattista Amati. Frequentist and bayesian approach to information retrieval. In Lalmas et al. [79], pages 13–24. ISBN 3-540-33347-9.

[6] Gianni Amati and C.J. van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trancactions on Information Systems*, 20(4):357–389, oct 2002.

[7] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. Business process execution language for web services (bpel4ws) version 1.1. `http://www-128.ibm.com/developerworks/library/specification/ws-bpel/`, May 2003.

[8] Association for Computing Machinery. The ACM Computing Classification Systems. `http://www.acm.org/class/1998`, 1998.

[9] Thomas Bailey and John Cowles. Cluster Definition by the Optimization of Simple Measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 1983.

[10] Richard E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, NJ, 1961.

[11] Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using Linear Algebra for Intelligent Information Retrieval. Technical Report UT-CS-94-270, Computer Science Department, dec 1994.

[12] J. C. Bezdek, W. Q. Li, Y. Attikiouzel, and M. Windham. A Geometric Approach to Cluster Validity for Normal Mixtures. *Soft Computing 1*, September 1997.

[13] J. C. Bezdek and N. R. Pal. Cluster Validation with Generalized Dunn's Indices. In N. Kasabov and G. Coghill, editors, *Proceedings of the 2nd international two-stream conference on ANNES*, pages 190–193, Piscataway, NJ, 1995. IEEE Press.

[14] Douglas Biber. The multidimensional approach to linguistic analyses of genre variation: An overview of methodology and findings. In *Computers and the Humanities*, volume 26, pages 331–345, 1992.

[15] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, 1997. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(97)00063-5.

[16] Elisabeth Sugar Boese and Adele E. Howe. Effects of Web Document Evolution on Genre Classification. In *Proceedings of the CIKM'05*. ACM Press, November 2005.

[17] Ivan Bretan, Johan Dewe, Anders Hallberg, and Niklas Wolkert. Web-specific genre visualization, 1999.

[18] A. Buja, D. F. Swayne, M. Littman, N. Dean, and H. Hofmann. XGvis: Interactive Data Visualization with Multidimensional Scaling. *Journal of Computational and Graphical Statistics*, 2001.

[19] WWW Consortium. OWL Web Ontology Language Semantics and Abstract Syntax. `http://www.w3.org/TR/owl-semantics/`, 2004.

[20] WWW Consortium. RDF Semantics. `http://www.w3.org/TR/rdf-mt/`, 2004.

[21] WWW Consortium. RDF/XML Syntax Specification. `http://www.w3.org/TR/rdf-syntax-grammar/`, 2004.

[22] W. Bruce Croft. A model of cluster searching based on classification. *Information Systems*, 5:189–195, 1980.

[23] Kevin Crowston and Marie Williams. The effects of linking on genres of web documents. In *HICSS*, 1999.

[24] Douglass R. Cutting, David R. Karger, and Jan O. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *SIGIR'93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 126–134, New York, NY, USA, 1993. ACM Press. ISBN 0-89791-605-0. doi: http://doi.acm.org/10.1145/160688.160706.

[25] Sanjoy Dasgupta. *Learning Probability Distributions*. PhD thesis, University of California at Berkeley, 2000.

[26] D.L. Davies and D.W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Learning*, 1(2), 1979.

[27] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[28] Simon Dennis. The sydney morning herald word database. `http://www2.psy.uq.edu.au/CogPsych/Noetica/OpenForumIssue4/SMH.html`, 1995.

[29] Simon Dennis, Peter Bruza, and Robert McArthur. Web searching: A process-oriented experimental study of three interactive search paradigms. *JASIST*, 53(2):120–133, 2002.

[30] Nigel Dewdney, Carol VanEss-Dykema, and Richard MacMillan. The form is the substance: Classification of genres in text. In *Proceedings of ACL Workshop on HumanLanguage Technology and Knowledge Management*, 2001.

[31] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD'01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-391-X. doi: http://doi.acm.org/10.1145/502512.502550.

[32] P. Diaconis and D. Freedman. Asymptotics of graphical projection pursuit. *Annals of Statistics*, 12:793–815, 1984.

[33] M. Dimitrova, A. Finn, N. Kushmerick, and B. Smyth. Web genre visualization. In *Proceedings of the Conference on Human Factors in Computing Systems*, 2002.

[34] Richard C. Dubes and Guangzhou Zeng. A test for spatial homogeneity in cluster analysis. *Journal of Classification*, 4:33–56, 1987.

[35] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000. ISBN 0471056693.

[36] J.C. Dunn. Well seperated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1974):95–104, 1974.

[37] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD96)*, 1996.

[38] B. S. Everitt. Cluster analysis. New York, Toronto, 1993.

[39] Aidan Finn and Nicholas Kushmerick. Learning to Classify Documents According to Genre. In *IJCAI-03 Workshop on Computational Approaches to Style Analysis and Synthesis*, 2003.

[40] K. Florek, J. Lukaszewiez, J. Perkal, H. Steinhaus, and S. Zubrzchi. Sur la liason et la division des points d'un ensemble fini. *Colloquium Methematicum*, 2, 1951.

[41] D. B. Fogel and L. J. Fogel. Special Issue on Evolutionary Computation. *IEEE Transaction of Neural Networks*, 1994.

[42] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78:553–569, 1983.

[43] W. B. Frakes. Term conflation for information retrieval. In *SIGIR'84: Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 383–389, Swinton, UK, 1984. British Computer Society. ISBN 0-521-26865-6.

[44] William B. Frakes and Christopher J. Fox. Strength and Similarity of Affix Removal Stemming Algorithms. *SIGIR Forum*, 37(1):26–30, 2003. ISSN 0163-5840. doi: http://doi.acm.org/10.1145/945546.945548.

[45] Norbert Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Trans. Inf. Syst.*, 7(3):183–204, 1989. ISSN 1046-8188. doi: http://doi.acm.org/10.1145/65943.65944.

[46] Norbert Fuhr. Probabilistic models in information retrieval. *Comput. J.*, 35(3):243–255, 1992.

[47] Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *ACM Trans. Inf. Syst.*, 9(3):223–248, 1991.

[48] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., 1998. ISBN 0-201-63361-2.

[49] Aristides Gionis, Alexander Hinneburg, Spiros Papadimitriou, and Panayiotis Tsaparas. Dimension induced clustering. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 51–60, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-135-X. doi: http://doi.acm.org/10.1145/1081870.1081880.

[50] A. Griffiths, H.C. Luckhurst, and P. Willett. Using inter-document similarity information in document retrieval systems. *Jounal of the American Society for Information Science*, 37:3–11, 1986.

[51] M.A. Hafer and S.F. Weiss. Word segmentation by letter successor varieties. *Information Processing & Management*, 10(11-12):371–386, 1974.

[52] Eui-Hong Han and George Karypis. Centroid-Based Document Classification: Analysis and Experimental Results. Technical Report 00-017, Univercity of Minnesota, Department of Computer Science / Army HPC Research Center, March 2000.

[53] Eui-Hong Han, George Karypis, and Vipin Kumar. Text Categorization Using Weight Adjusted k-Nearest Neighbor Classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 53–65, Minneapolis, USA, 2001. Univercity of Minnesota, Department of Computer Science / Army HPC Research Center.

[54] S.P. Harter. A probabilistic approach to automatic keyword indexing. part i: On the distribution of specialty words in a technical literature. *J.ASIS*, 26, 1975.

[55] S.P. Harter. A probabilistic approach to automatic keyword indexing. part ii: An algorithm for probabilistic indexing. *J.ASIS*, 26, 1975.

[56] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *SIGIR'96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84, New York, NY, USA, 1996. ACM Press. ISBN 0-89791-792-8. doi: http://doi.acm.org/10.1145/243199.243216.

[57] Alexander Hinneburg and Daniel A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 506–517, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-615-7.

[58] Martin Hitz, Gerti Kappel, Elisabeth Kapsammer, and Werner Retschitzegger. *UML @ Work*. dpunkt.verlag, 2005. ISBN 3-89864-261-5.

[59] Andreas Hotho. *Clustern mit Hintergrundwissen*, volume 286 of *Diski*. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2004. ISBN 3-89838-286-9. URL `http://www.kde.cs.uni-kassel.de/hotho/pub/2004/dissAho.pdf`.

[60] L. J. Hubert and J. Schultz. Quadratic assignment as a general data-analysis strategy. *British Journal of Mathematical and Statistical Psychology*, 29:190–241, 1976.

[61] Georgiana Ifrim, Martin Theobald, and Gerhard Weikum. Learning Word-to-Concept Mappings for Automatic Text Classification. In *Proceedings of the ICML-Learning in Web Search Workshop*, 2005.

[62] Makoto Iwayama and Takenobu Tokunaga. Cluster-based text categorization: a comparison of category search strategies. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *SIGIR'95: Proceedings of the 18th ACM International Conference on Research and Development in Information Retrieval*, pages 273–281, Seattle, USA, 1995. ACM Press, New York, US.

[63] A. K. Jain and J. V. Moreau. Bootstrap technique in cluster analysis. *Pattern Recognition*, 20(5):547–568, 1987.

[64] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering in Data*. Prentice Hall, Englewood Cliffs, NJ, 1990. ISBN 0-13-022278-X.

[65] Kurt Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.*, volume 1 of *Monographs in Theoretical Computer Science*. Springer-Verlag, 1997. ISBN 3-540-60943-1.

[66] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant Features and the Subset Selection Problem. In William W. Cohen and Haym Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, San Francisco, CA, 1994. Morgan Kaufmann.

[67] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32, 1967.

[68] Jussi Karlgren and Douglass Cutting. Recognizing text genres with simple metrics using discriminant analysis. In *Proceedings of the 15th. International Conference on Computational Linguistics (*COLING 94*)*, volume II, pages 1071 – 1075, Kyoto, Japan, 1994.

[69] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. Technical Report Paper No. 432, University of Minnesota, Minneapolis, 1999.

[70] George Karypis and Eui-Hong Han. Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval and categorization. Technical report tr-00-0016, University of Minnesota, 2000. URL `citeseer.ist.psu.edu/karypis00concept.html`.

[71] Leonard Kaufman and Peter J. Rousseuw. *Finding Groups in Data*. Wiley, 1990.

[72] Brett Kessler, Geoffrey Nunberg, and Hinrich Schütze. Automatic detection of text genre. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 32–38, Somerset, New Jersey, 1997. Association for Computational Linguistics.

[73] R. W. Klein and R. C. Dubes. Experiments in Projection and Clustering by Simulated Annealing. *Pattern Recognition*, 22:213–220, 1989.

[74] T. Kohonen. *Self Organization and Assoziative Memory*. Springer, 1990.

[75] T. Kohonen, S. Kaski, K. Lagus, J. Salojrvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. In *IEEE Transactions on Neural Networks*, volume 11, may 2000.

[76] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *International Conference on Machine Learning*, pages 284–292, 1996. URL `citeseer.ist.psu.edu/koller96toward.html`.

[77] J. B. Kruskal. Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis. *Psychometrika*, 29(1), March 1964.

[78] Krishna Kummamuru, Rohit Lotlikar, Shourya Roy, Karan Singal, and Raghu Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*, pages 658–665, 2004.

[79] Mounia Lalmas, Andy MacFarlane, Stefan M. Rüger, Anastasios Tombros, Theodora Tsikrika, and Alexei Yavlinsky, editors. *Advances in Information Retrieval, 28th European Conference on IR Research, ECIR 2006, London, UK, April 10-12, 2006, Proceedings*, volume 3936 of *Lecture Notes in Computer Science*, 2006. Springer. ISBN 3-540-33347-9.

[80] Dawn Lawrie, W. Bruce Croft, and Arnold L. Rosenberg. Finding topic words for hierarchical summarization. In *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 349–357, 2001.

[81] Dawn J. Lawrie. *Language Models for Hierarchical Summarization.* PhD thesis, University of Massachusetts at Amherst, 2003b.

[82] Dawn J. Lawrie and W. Bruce Croft. Generating hierarchical summaries for web searches. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, pages 457–458, 2003.

[83] Yong-Bae Lee and Sung Hyon Myaeng. Text genre classification with genre-revealing and subject-revealing features. In *Proc. 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 145–150. ACM Press, 2002. ISBN 1-58113-561-0. doi: http://doi.acm.org/10.1145/564376.564403.

[84] Thomas Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* John Wiley & Sons, New York, 1990.

[85] V. I. Levenshtein. Binary codes capable of correcting deletions insertions and reversals. *ISov Phys Dokl*, 6:707–710, 1966.

[86] Erel Levine and Eytan Domany. Resampling method for unsupervised estimation of cluster validity. *Neural Computation*, 13:2573–2593, 2001.

[87] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1987.

[88] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[89] Y. Matsuo and M. Ishizuka. Keyword Extraction from a Single Document using Word Co-ocurrence Statistical Information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169, 2004.

[90] Sven Meyer zu Eißen and Benno Stein. Analysis of Clustering Algorithms for Web-based Search. In Dimitris Karagiannis and Ulrich Reimer, editors, *Practical Aspects of Knowledge Management*, volume 2569 LNAI of *Lecture Notes in Artificial Intelligence*, pages 168–178. Springer, December 2002. ISBN 3-540-00314-2.

[91] Sven Meyer zu Eißen and Benno Stein. The AIsearch Meta Search Engine Prototype. In Amit Basu and Soumitra Dutta, editors, *Proceedings of the 12th Workshop on Information Technology and Systems (WITS 02), Barcelona Spain.* Technical University of Barcelona, December 2002.

[92] Sven Meyer zu Eißen and Benno Stein. Genre Classification of Web Pages: User Study and Feasibility Analysis. In Susanne Biundo, Thom Frühwirth, and Günther Palm, editors, *KI 2004: Advances in Artificial Intelligence*, volume 3228 LNAI of *Lecture Notes in Artificial Intelligence*, pages 256–269, Berlin Heidelberg New York, September 2004. Springer. ISBN 0302-9743.

[93] Sven Meyer zu Eißen and Benno Stein. Wrapper Generation with Patricia Trees. In Benno Stein, Sven Meyer zu Eißen, and Andreas Nürnberger, editors, *Machine Learning and Interaction for Text-Based Information Retrieval (TIR 04)*, Workshop Proceedings, pages 69–76. Universität Ulm, September 2004.

[94] Sven Meyer zu Eissen and Benno Stein. Intrinsic plagiarism detection. In Mounia Lalmas, Andy MacFarlane, Stefan M. Rüger, Anastasios Tombros, Theodora Tsikrika, and Alexei Yavlinsky, editors, *Proceedings of the European Conference on Information Retrieval (ECIR 2006)*, volume 3936 of *Lecture Notes in Computer Science*, pages 565–569. Springer, 2006. ISBN 3-540-33347-9.

[95] Sven Meyer zu Eißen and Benno Stein. Realization of web-based simulation services. *CiI – Computers in Industry: Special Issue on Advanced Computer Support of Engineering and Service Processes of Virtual Enterprises*, 57 (3):261–271, 2006. ISSN 0166-3615. doi: http://dx.doi.org/10.1016/j.compind.2005.12.007.

[96] Sven Meyer zu Eißen, Benno Stein, and Martin Potthast. The Suffix Tree Document Model Revisited. In Klaus Tochtermann and Hermann Maurer, editors, *Proceedings of the 5th International Conference on Knowledge Management (I-KNOW 05), Graz*, Journal of Universal Computer Science, pages 596–603. Know-Center, July 2005.

[97] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill Higher Education, 1997. ISBN 0070428077.

[98] Object Management Group (OMG). Model driven architecture (mda) guide. `http://www.omg.org/docs/omg/03-06-01.pdf`, 2003.

[99] Object Management Group (OMG). The uml metamodel. `http://www.omg.org/cgi-bin/doc?ptc/2004-10-05`, 2003.

[100] Object Management Group (OMG). The unified modeling language (UML) specification, version 2. `http://www.uml.org`, 2005.

[101] Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: a probabilistic analysis. In

PODS'98: Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, pages 159–168, New York, NY, USA, 1998. ACM Press. ISBN 0-89791-996-3. doi: http://doi.acm.org/10.1145/275487.275505.

[102] Carl Adam Petri. *Kommunikation mit Automaten.* PhD thesis, University of Bonn, 1962.

[103] Peter Pirolli, Patricia Schank, Marti Hearst, and Christine Diehl. Scatter/gather browsing communicates the topic structure of a very large text collection. In *CHI'96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–220, New York, NY, USA, 1996. ACM Press. ISBN 0-89791-777-4. doi: http://doi.acm.org/10.1145/238386.238489.

[104] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR'98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM Press. ISBN 1-58113-015-5. doi: http://doi.acm.org/10.1145/290941.291008.

[105] Alexandrin Popescul and Lyle H. Ungar. Automatic Labeling of Document Clusters. `http://citeseer.nj.nec.com/popescul00automatic.html`, 2000.

[106] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

[107] Martin Porter. Snowball. `http://snowball.tartarus.org/`, 2001.

[108] V. V. Raghavan and K. Birchand. A Clustering Strategy Based on a Formalism of the Reproduction Process in a Natural System. In *Proceedings of the Second International Conference on Information Storage and Retrieval*, pages 10–22, 1979.

[109] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

[110] Georg Rehm. Towards Automatic Web Genre Identification. In *Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS'02)*. IEEE Computer Society, January 2002.

[111] C. J. van Rijsbergen. *Information Retrieval*. Buttersworth, London, 1979.

[112] S. E. Robertson. *The probability ranking principle in IR*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. ISBN 1-55860-454-5.

[113] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X.

[114] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *CIKM'04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-874-1. doi: http://doi.acm.org/10.1145/1031171.1031181.

[115] T.G. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1 - From Yesterday's News to Tomorrow's Language Resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, 2002.

[116] Dmitri Roussinov, Kevin Crowston, Mike Nilan, Barbara Kwasnik, Jin Cai, and Xiaoyong Liu. Genre based navigation on the web. In *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2001.

[117] Tom Roxborough and Arunabha. Graph Clustering using Multiway Ratio Cut. In Stephen North, editor, *Graph Drawing*, Lecture Notes in Computer Science, Springer, 1996.

[118] Reinhard Sablowski and Arne Frick. Automatic Graph Clustering. In Stephan North, editor, *Graph Drawing*, Lecture Notes in Computer Science, Springer, 1996.

[119] G. Salton. *Cluster search strategies and the optimization of retrieval effectiveness.* Prentice-Hall, Englewood Cliffs, N.J., 1971.

[120] G. Salton and M. J. Gill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

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

[122] Mark Sanderson and Bruce Croft. Deriving concept hierarchies from text. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–213, New York, NY, USA, 1999. ACM Press. ISBN 1-58113-096-1. doi: http://doi.acm.org/10.1145/312624.312679.

[123] Marina Santini. Common criteria for genre classification: Annotation and granularity. In *Proceedings of the ECAI-Workshop TIR-06*, Riva del Garda, Italy, 2006.

[124] Claude E. Shannon. A Mathematical Theory of Communication. *Bell Sys. Tech. J.*, 27:379–423, 623–656, 1948.

[125] P. H. A. Sneath. The application of computers to taxonomy. *J. Gen. Microbiol.*, 17, 1957.

[126] Karen Sparck-Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[127] Amanda Spink, Judy Bateman, and Bernard J. Jansen. Users' Searching Behavior On The Excite Web Search Engine. In Hermann A. Maurer and Richard G. Olson, editors, *Proceedings of WebNet 98 - World Conference on the WWW and Internet & Intranet*, Orlando, Florida, USA, nov 1998. ISBN 1-880094-31-2.

[128] E. Stamatatos, N. Fakotakis, and G. Kokkinakis. Text genre detection using common word frequencies. In *Proceedings of the 18th Int. Conference on Computational Linguistics*, Saarbrücken, Germany, 2000.

[129] Benno Stein. Readings in Advanced Web Technology. http://www.uni-weimar.de/m/webis, 2006.

[130] Benno Stein and Sven Meyer zu Eißen. Document Categorization with MajorClust. In Amit Basu and Soumitra Dutta, editors, *Proceedings of the 12th Workshop on Information Technology and Systems (WITS 02), Barcelona Spain*, pages 91–96. Technical University of Barcelona, December 2002.

[131] Benno Stein and Sven Meyer zu Eißen. Automatic Document Categorization: Interpreting the Perfomance of Clustering Algorithms. In Andreas Günter, Rudolf Kruse, and Bernd Neumann, editors, *KI 2003: Advances in Artificial Intelligence*, volume 2821 LNAI of *Lecture Notes in Artificial Intelligence*, pages 254–266. Springer, September 2003. ISBN 3-540-20059-2.

[132] Benno Stein and Sven Meyer zu Eißen. The AIsearch Meta Search Engine Homepage. `http://www.aisearch.de`, 2003-2007.

[133] Benno Stein and Sven Meyer zu Eißen. Automatische Kategorisierung für Web-basierte Suche: Einführung, Techniken und Projekte. *KI – Künstliche Intelligenz: Special Issue on Adaptive Multimedia Retrieval*, 4: 11–17, November 2004. ISSN 0933-1875.

[134] Benno Stein and Sven Meyer zu Eißen. Topic Identification: Framework and Application. In Klaus Tochtermann and Hermann Maurer, editors, *Proceedings of the 4th International Conference on Knowledge Management (I-KNOW 04), Graz, Austria*, Journal of Universal Computer Science, pages 353–360, Graz, Austria, July 2004. Know-Center.

[135] Benno Stein and Sven Meyer zu Eißen. Near Similarity Search and Plagiarism Analysis. In M. Spiliopoulou, R. Kruse, C. Borgelt, A. Nürnberger, and W. Gaul, editors, *From Data and Information Analysis to Knowledge Engineering*, pages 430–437. Springer, 2006. ISBN 1431-8814.

[136] Benno Stein, Sven Meyer zu Eißen, Gernot Gräfe, and Frank Wissbrock. Automating Market Forecast Summarization from Internet Data. In Pedro Isaías and Miguel Baptista Nunes, editors, *Fourth International Conference on WWW/Internet, Lissabon*, pages 395–402. IADIS Press, October 2005. ISBN 972-8924-02-X.

[137] Benno Stein, Sven Meyer zu Eißen, and Martin Potthast. Syntax versus Semantics: Analysis of Enriched Vector Space Models. In Benno Stein and Odej Kao, editors, *Third International Workshop on Text-Based Information Retrieval (TIR 06)*, pages 47–52. University of Trento, Italy, August 2006.

[138] Benno Stein, Sven Meyer zu Eißen, and Frank Wißbrock. On Cluster Validity and the Information Need of Users. In M. H. Hanza, editor, *Proceedings of the 3rd IASTED International Conference on Artificial Intelligence and Applications (AIA 03), Benalmádena, Spain*, pages 216–221, Anaheim, Calgary, Zurich, September 2003. ACTA Press. ISBN 0-88986-390-3.

[139] Benno Stein and Oliver Niggemann. On the Nature of Structure and its Identification. In Peter Widmayer, Gabriele Neyer, and Stefan Eidenbenz, editors, *Graph-Theoretic Concepts in Computer Science*, volume 1665 LNCS of *Lecture Notes in Computer Science*, pages 122–134. Springer, June 1999. ISBN 3-540-66731-8.

[140] Benno Stein and Martin Potthast. Putting Successor Variety Stemming to Work. In *Advances in Data Analysis (to appear)*. Springer, 2007.

[141] Alexander Strehl. *Relationship-based Clustering and Visualization for High-dimensional Data Mining*. PhD thesis, University of Texas at Austin, 2002.

[142] Sun Microsystems. Java Architecture for XML Binding (JAXB Specification). `http://java.sun.com/xml/downloads/jaxb.html`, 2003.

[143] Jürgen Teich. *Digitale Hardware/Software-Systeme*. Springer, 1997.

[144] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters via the gap statistic. *Journal of Royal Statistics and Social Behaviour*, 63 (2):411–423, 2001.

[145] Myron Tribus. *Thermostatics and Thermodynamics*. Princeton, N.J., Van Nostrand, 1961.

[146] Yuen-Hsien Tseng. Multilingual keyword extraction for term suggestion. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR*

Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia, pages 377–378, 1998.

[147] P. Turney. Coherent keyphrase extraction via web mining. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 434–439, aug 2003.

[148] University of Leipzig. Wortschatz. `http://wortschatz.uni-leipzig.de`, 1995.

[149] University of Stuttgart. The decision tree tagger. `http://www.ims.uni-stuttgart.de`, 1996.

[150] Vishwa Vinay, Ingemar J. Cox, Natasa Milic-Frayling, and Ken Wood. Measuring the complexity of a collection of documents. In Lalmas et al. [79], pages 107–118. ISBN 3-540-33347-9.

[151] Vivisimo. The Vivisimo Meta Search Engine Homepage. `http://www.vivisimo.com`, 2000-2006.

[152] Ulrike von Luxburg. *Statistical Learning with Similarity and Dissimilarity Functions*. PhD thesis, Technische Universität Berlin, 2004.

[153] Ellen M. Voorhees. The cluster hypothesis revisited. In *SIGIR'85: Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 188–196, New York, NY, USA, 1985. ACM Press. ISBN 0-89791-159-8. doi: http://doi.acm.org/10.1145/253495.253524.

[154] Wikipedia. Definition of Surprisal. `http://en.wikipedia.org/wiki/Surprisal`, 2006.

[155] Peter Willett. Recent trends in hierarchical document clustering. *Information Processing & management*, 24(5):577–597, 1988.

[156] Joachim Winter. Definition of bounded rationality. `http://www.sfb504.uni-mannheim.de/glossary/bounded.htm`, 1999.

[157] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. KEA: Practical automatic keyphrase extraction. In *ACM DL*, pages 254–255, 1999.

[158] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, November 1993.

[159] Wei Xu and Yihong Gong. Document clustering by concept factorization. In *SIGIR'04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 202–209, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-881-4. doi: http://doi.acm.org/10.1145/1008992.1009029.

[160] J. T. Yan and P. Y. Hsiao. A fuzzy clustering algorithm for graph bisection. *Information Processing Letters*, 52, 1994.

[161] David Yarowsky. Word-sense disambiguation using statistical models of roget's categories trained on large corpora. In *COLING*, pages 454–460, 1992.

[162] C. T. Zahn. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on computers*, C-20(1), 1971.

[163] Oren Zamir and Oren Etzioni. Web Document Clustering: A Feasibility Demonstration. In *SIGIR'98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 46–54, University of Washington, Seattle, USA, 1998.