Language Models and Smoothing Methods for Information Retrieval

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Informatik und angewandte Kognitionswissenschaft,
der Universität Duisburg-Essen
zur Erlangung des akademischen Grades
eines Doktors der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Dissertation

von

Najeeb A. Abdulmutalib M.Sc
aus Ben Walid (Libya)

Gutachter:
Prof. Dr.-Ing. Norbert Fuhr
Prof. Dr.-Ing. Gerhard Weikum

Tag der mündlichen Prüfung: 29. Oktober 2010

Abstract

Designing an effective retrieval model that can rank documents accurately for a given query has been a central problem in information retrieval for several decades. An optimal retrieval model that is both effective and efficient and that can learn from feedback information over time is needed. Language models are new generation of retrieval models and have been applied since the last ten years to solve many different information retrieval problems. Compared with the traditional models such as the vector space model, they can be more easily adapted to model non traditional and complex retrieval problems and empirically they tend to achieve comparable or better performance than the traditional models. Developing new language models is currently an active research area in information retrieval.

In the first stage of this thesis we present a new language model based on an odds formula, which explicitly incorporates document length as a parameter.

To address the problem of data sparsity where there is rarely enough data to accurately estimate the parameters of a language model, smoothing gives a way to combine less specific, more accurate information with more specific, but noisier data. We introduce a new smoothing method called exponential smoothing, which can be combined with most language models. We present experimental results for various language models and smoothing methods on a collection with large document length variation, and show that our new methods compare favourably with the best approaches known so far.
We discuss the collection effect on the retrieval function, where we investigate the performance of well known models and compare the results conducted using two variant collections.

In the second stage we extend the current model from flat text retrieval to XML retrieval since there is a need for content-oriented XML retrieval systems that can efficiently and effectively store, search and retrieve information from XML document collections. Compared to traditional information retrieval, where whole documents are usually indexed and retrieved as single complete units, information retrieval from XML documents creates additional retrieval challenges. By exploiting the logical document structure, XML allows for more focussed retrieval that identifies elements rather than documents as answers to user queries.

Finally we show how smoothing plays a role very similar to that of the *idf* function: beside the obvious role of smoothing, it also improves the accuracy of the estimated language model. The within document frequency and the collection frequency of a term actually influence the probability of relevance, which led us to a new class of smoothing function based on numeric prediction, which we call empirical smoothing. Its retrieval quality outperforms that of other smoothing methods.

Dedication

In memory of my father Abdullah Abdulmutalib
to my mother, my wife and my children

Acknowledgements

All praise is due to Allah Almighty, the most Beneficent and Merciful, who created me and guides me. The one who induced man with intelligence, knowledge and wisdom.

First, I would like to take this opportunity to express the deepest appreciation to my supervisor Professor Norbert Fuhr for his invaluable support, encouragement, supervision and useful suggestions throughout this research work. Without his guidance and persistent help this thesis would not have been possible.

I sincerely thank all of our group colleagues for their help and the friendly atmosphere that I enjoyed during all the period of my study. My sincere thanks and appreciations to Sascha Kriewel for his useful suggestions which was very helpful in improving this thesis. My special thanks to our brilliant colleague Henrik Nottleman (Late), who left us too early.

I would especially thank my mother and sisters for their never-ending love and support throughout my life.

I also want to express my deepest gratitude to my uncle Saad, haj Mansur and uncle Jamal for their support and encouragement.

Last but not least, I would like to thank my wife Sumaya for her sincere, endless support and her infinite patience which can not be measured.

Najeeb A. Abdulmutalib
Duisburg, July 2010

Contents

1

Introduction

1.1 Information Retrieval

Information retrieval (IR) is the process of locating information that fits a user's information need, which is usually expressed as a search query. It requires a well-defined model of how users search for information (their search behavior), and an understanding of the meaning of both a document and a user's query. Modeling a user's search behavior is important because an IR technique's effectiveness will differ according to how the user searches. Factors such as what the user is searching for, the corpora of text being searched, and the user's interaction with the system all affect an IR system's success in retrieving relevant documents. A good IR system is therefore a combination of effective retrieval algorithms operating according to an appropriate model of user behavior.

We are increasingly turning to data that is digitally stored, published, and transmitted in increasing volumes. The amount of information has increased, and so has its reach. Growth in the number of readers and publishers changes the way information is used, making it more difficult to model users' search behaviors, and to "fit" users' needs to retrieval information.

The early age of IR began in the mid-1940 and lasted until the 1960s. During this time, researchers developed the concepts of recall and precision, free-text search, and thesauri such as the Medical Subject Headings (MeSH) and the Inspec Thesaurus, which evolved to standardize nomenclature [Lesk (1995)]. However, IR systems in use, and machine-readable text of any sizable volume simply did not exist. In 1970s, IR saw some maturation with commercial IR systems becoming a reality as the computer became widespread, and word-processing produced that a large body of machine-readable text. IR was slowly extending its reach to cover more of society, though only in limited communities, and still required expertise to use the systems.

In the 1980s, several online database systems were developed, offering full text, indexes, and abstracts. Storage costs plummeted with the advent of the CD-ROM, and as word processing became more ubiquitous, machine-readable information exploded. Thus, IR systems began to be used in an increasing number of places, extending searching from the scientific to the general business community. In the lab, developments focused on the use of AI techniques, including expert systems and natural language programming.

In the 1990s, expert systems' limitations became apparent, and attention shifted to newer AI techniques, such as neural networks and genetic algorithms [Baldi et al. (1997)]. However, most text in this age was written on computers and then simply printed. Desktop PCs and information started to be everywhere and the desktop word processor and database usually installed on PCs were used solely by the PC owner. The average user simply did not require sophisticated retrieval. The Internet, and the World Wide Web in particular, changed this situation completely, but the techniques that were developed from the IR community were ill-prepared for the volume of information that was about to be generated.

The Web enabled people to access and, crucially, to provide information. With machine-readable text being produced at prodigious rate, everyone suddenly needed the ability to search for information. The field of IR had thus matured from its early years of research and science, through its adolescent experimenting with the business community, to reach all of online society. The Web has brought every area of society access to unstructured information by means of its key application, the web search engine.

From the beginning, users needed to search the Web, and the first search engines - the World Wide Web Worm and JumpStation - arose almost immediately to meet that need. Released in 1993, just 10 months after the seminal Mosaic browser brought the Web to the masses, these crude systems mostly ignored existing IR techniques. JumpStation, for example, indexed titles and headers and stored the results in a database that it searched linearly and was the first search engine to use anchor text (the text surrounding a hyperlink) to facilitate retrieval. However, both systems relied on simple keyword search and listed pages in the order they were retrieved from the database. Yahoo! was released a year later as a searchable directory, but it was equally unsophisticated because it employed human crawlers to enter, classify, and describe each page manually, and could perform keyword search only on page descriptions rather than content. Later in 1994 came WebCrawler, which was the first search engine to employ full-text search for every Web page. Soon after, developers began using more complex IR techniques in more sophisticated engines. Lycos, for example, was able to rank its results according to relevance, and AltaVista introduced a crude form of natural language querying with Boolean operators to the Web with its December 1995 release. However, as the size of the Web's information increased, the problem's scale changed. The technology required to find and index all of this information, and then return it as quickly as possible, presented substantial challenges, leaving no computational horsepower to implement more sophisticated retrieval techniques. Competition soon focused on who could index the most pages, and relevance was forgotten as retrieval criterion. The Web became increasingly less useful, and by 1997, only one of the top four search engines could actually find itself [Brin and Page (1998)]. Sophisticated IR techniques did not cope well with the Web's search environment because it was completely different from the environments for which they were designed. Many algorithms, such as the standard vector space model, assume a different model of search behavior, expecting users to enter verbose queries whereas Web users' queries tend to be short. (These algorithms tend to retrieve documents that contain nothing more than the user's query plus one or two extra terms [Brin and Page

(1998)]). Other techniques, such as latent semantic indexing, simply can not scale the Web's volume of information.

On the other hand, comparing search engines with modern IR techniques is difficult because Web search works only in an environment in which sophisticated IR techniques work very poorly. Numerous experiments have been conducted, but the results reflect more of the differences in search environments that each must work in than on the effectiveness of the techniques employed.

The Text Retrieval Conference (TREC) [1], measuring the search engines' effectiveness, [Hawking and Craswell (2005)] illustrated how much Web search differs from traditional IR, not just in terms of the quantity and quality of information but also in the user's needs and expectations. Specifically, the differences between the two environments have been identified as data set size, document type, interlinking between documents, volume of queries submitted (around 500 million queries per day on the Web), length of typical queries (two words, on average for the Web) and types of search activity undertaken (IR users expert to retrieve relevant text items, Web users expert to retrieve pointers to relevant sites). TREC has concluded that hyperlink and other Web evidence is highly valuable for some types of search task, but not for others [Hawking et al. (2001)].

The field of information retrieval has been primarily concerned with developing algorithms to identify relevant pieces of information in response to a user's information need. Much research in the area has focused on developing formal models of relevance: [Robertson (1977)], [Robertson and Walker (1994)], [Robertson and Sparck Jones (1976)], [Turtle and Croft (1991)], Van Rijsbergen's logical implication model, [van Rijsbergen (1986)] viewed relevant documents as those from which the query q could be inferred using a probabilistic logic (written $d \rightarrow q$). [Turtle and Croft (1990)], [Turtle and Croft (1991)] implemented a model called inference network model in the INQUERY system based on calculating $P(I|d)$ the probability that an information need I is satisfied given a document.

More recently, in the language modeling approach, [Ponte and Croft (1998b)], [Berger and Lafferty (1999)], [Miller et al. (1999)], [Song and Croft (1999)], [Hiemstra (2001)] documents are ranked by $P(q|d)$, the probability of generating a query text q given an estimated language model for document d. The advantage of this approach was to shift from developing heuristic $tf \cdot idf$ weights for representing term importance to instead focus on estimation techniques for the document model.

1.2 Motivations and main issues

The language modeling approach to IR takes a generational view of the retrieval process. A query is seen as an utterance made by the user of the IR system in an attempt to describe an ideal document that he wants to retrieve. The task of document retrieval becomes that of modeling this special type of language generation process. The notion of being *relevant* is rephrased as being such a document that can, when conceptualized as an ideal document by the user of the IR system, induce him to generate the given query. The IR system ranks each document by the likelihood that the document is

[1] http://trec.nist.gov/

the one the user intended to describe by the query. So the retrieval problem can be simplified by phrasing it as a generative language modeling problem and the resulting approach will provide effective retrieval. Since the first publication of the application of statistical language modeling for IR in 1998, many different variants have been proposed, based on e.g., likelihood ratio, Kullback/Leibler divergence, query likelihood and document likelihood. We have studied the properties of these variations and their relationships and discuss the various alternatives and would like to add new effective models. The motivation of our work is:

- We want to study the document length effect on the retrieval effectiveness and compare with the traditional models.
- Further discussion of the language modeling approach in general, and in particular to develop a new effective language model based on an odds formula and a new smoothing method.
- The increasing availability of XML collections offers the opportunity for developing appropriate retrieval methods.
- We study the connection between smoothing and $tf \cdot idf$ weighting and the role that a term's relative collection frequency P_{avg} plays beside smoothing.

1.3 Structure of this dissertation

The remainder of this thesis is organized as follows:

Chapter 2: Probabilistic IR

In this chapter an introduction to and a survey on probabilistic information retrieval (IR) are given.

Chapter 3: Language Models and Smoothing Methods

Here we will introduce the language modeling approaches in information retrieval.
The effect of document length on the retrieval effectiveness is also discussed in this chapter.

Chapter 4: XML retrieval

XML retrieval is an important area for the application of IR methods. Here we focus on XML concepts that are of concern to retrieval.

Chapter 5: Relevance and Evaluation

This chapter describes the relevance and evaluation concepts in information retrieval. Also a list of the most common test collections is discussed. We focus particularly on test collections for ad hoc information retrieval system evaluation.

Chapter 6: Experiments with Divergence From Randomness (DFR)

DFR is a well known probabilistic model which infers the informativeness of a term by the divergence between its distribution in the top ranked documents and a random distribution. Experiments with DFR are discussed in this chapter.

Chapter 7: Language Models and Smoothing for Collections with Large Variation in Document Length

Here we present a new language model based on an odds formula, and a new smoothing method called exponential smoothing, which can be combined with most language models.

Chapter 8: Using the Language Model for XML retrieval

In this chapter, we present an extension of the odds language model to use it in structured document retrieval according to the specificity dimension.

Chapter 9: Empirical smoothing

Here we study the connection between smoothing and $tf \cdot idf$ weighting and the role that P_{avg} plays beside smoothing and we present a new smoothing technique with linear regression using empirical tests.

Chapter 10: Conclusion and Outlook

The conclusions drawn from the overall thesis and the avenues for future work are described in this chapter.

2
Probabilistic IR

This chapter surveys probabilistic approaches to modeling information retrieval.

The first probabilistic model for information retrieval, namely the probabilistic indexing model of Maron and Kuhns [Maron and Kuhns (1960)] is based on the idea of query generation. The model intends to infer the probability that a document is relevant to a query based on the probability that a user who likes the document would have used this query, but the model suffered from the difficulty of parameter estimation.
In the classical probabilistic approach to information retrieval [Robertson and Sparck Jones (1976)] two models are estimated for each query, one modeling relevant documents, the other modeling non-relevant documents.
The probability ranking principle (PRP) [Robertson (1977)] says that optimum retrieval is achieved when documents are ranked according to decreasing values of the probability of relevance (with respect to the current query).

2.1 Relevance oriented models

The goal of relevance-oriented search is to estimate the probability of relevance of a document for a given query.

2.1.1 Binary Independence Retrieval model

The basic idea of the binary independence retrieval (BIR) model [Robertson and Sparck Jones (1976)], [van Rijsbergen (1977)] is that term distributions are different for relevant and non relevant documents. The basic BIR model only regards term presence or absence, so every document d can be described with a term set d^T. Computing the odds of relevance of a document and applying Bayes' theorem [Kraaij (2004)], the formula which estimates the probability that documents described by a set d^T are relevant for a certain query using the terms q^T can be derived by the ranking formula:

$$w_i = \sum_{t_i \in q^T \cap d^T} \log \frac{p_i(1-q_i)}{q_i(1-p_i)} \tag{2.1}$$

This basic BIR model can only be applied after estimating the parameters p_i and q_i for all query terms, e.g., for each term we have to estimate the probability that this term occurs in a relevant document (p_i) and in a non relevant document (q_i). [Robertson and Sparck Jones (1976)] discuss four methods to estimate these parameters. There are n_i documents that contain the query term, there are R documents which are relevant for the query and the number of relevant documents that contains the query terms is r_i. They suppose that the distribution of terms is independent both in the set of relevant documents and in the set of irrelevant documents. Assuming full relevance information and that probability of relevance is based on both presence and absence of query terms in documents, p_i can be estimated by r_i/R and q_i by $(n_i - r_i)/(N - R)$. Substitution into the individual term weight of (2.1) yields:

$$w_i = \log \frac{r_i/(R-r_i)}{(n_i - r_i)/((N-n_i)-(R-r_i))} \tag{2.2}$$

The probabilistic relevance models presuppose full relevance information. The formula 2.1 is undefined if there is no relevance information ($R = r = 0$). Therefore a small constant (0.5) is added to both numerator and denominator of both probability estimates. and thus without relevance information 2.2 can be rewritten as 2.3 which is in fact an inverse document frequency (*idf*) weight:

$$w_i = \log \frac{N - n_i + 0.5}{n_i + 0.5} \tag{2.3}$$

In fact, the BIR model is closely related to the Naive Bayes classifier, which is often used for (supervised) text classification [Lewis (1998)]. Without relevance information, the BIR model is very weak in comparison with standard $tf \cdot idf$ since it lacks a term frequency component as well as document length normalization.

2.1.2 The binary independence indexing model

The binary independence indexing (BII) model [Fuhr and Buckley (1991b)] is a variant of the very first probabilistic IR model, namely the indexing model of [Maron and Kuhns (1960)]. Whereas the BIR model regards a single query w.r.t. a number of documents, the BII model observes one document in relation to a number of queries submitted to the system. The final BII formula yields:

$$P(R|q_k, d_m) = c_k \cdot P(R|d_m) \cdot \prod_{t_i \in q_k^T \cap d_m^T} \frac{P(R|t_i, d_m)}{P(R|d_m)} \tag{2.4}$$

Here T is a set of terms $t_1..., t_n$, the query q_k is a subset $q_k^T \subset T$, d_m^T is the set of terms occurring in the document, $P(R|d_m)$ is the probability that document d_m will be judged relevant to an arbitrary request, c_k is a constant for a given query q_k and $P(R|t_i, d_m)$ is the probabilistic index term weight of t_i w.r.t. d_m, the probability that document d_m will be judged relevant to an arbitrary query, given that it contains t_i.

The basic idea for the new approach stems from the Darmstadt Indexing Approach (DIA) [Biebricher et al. (1988)],[Fuhr (1989a)], where in DIA, the indexing task is subdivided in a description step and a decision step. In the description step, relevance descriptions for term-document pairs (t_i, d_m) are formed. A relevance description comprises a set of features that are considered to be important for the task of assigning weights to terms w.r.t. documents. So a relevance description $x(t_i, d_m)$ contains values of attributes of the term t_i, the document d_m and their relationship.

Then machine learning methods are used for estimating $P\left(R|x(t_i, d_m)\right)$. As an alternative method one can assume a document to consist of independent components (e.g. sentences or words) to which the indexing weights relate, but experimental evaluations showed only moderate results for this approach [Kwok (1990)].

2.1.3 The 2-Poisson model

The 2-Poisson Model [Bookstein and Swanson (1975), Harter (1975a), Harter (1975b)] assumed that a collection of documents can be divided in two classes. A document is either about a certain term (elite) or not (non-elite). Both document classes are modeled by a Poisson distribution, but the probability of a term i occurring k times is in this case modeled by combining the estimates from both models:

$$P(k; \lambda_i, \mu_i) = \alpha \cdot \frac{\lambda_i^k}{k_!} \cdot \mathrm{e}^{\lambda_i} + (1-\alpha)\frac{\mu_i^k}{k_!} \cdot \mathrm{e}^{\mu_i} \tag{2.5}$$

where λ_i and μ_i are the average numbers of occurrences in the class of elite and non-elite documents respectively, α is the probability that a document belongs to the elite set. The 2-Poisson model postulates that a word can either be of central importance for the content of a document, or occurs spuriously and should not be considered as an index term. A new probabilistic model based on the 2-Poisson distribution was developed by [Robertson and Walker (1994)] .

2.1.4 $tf \cdot idf$ and BM25

The $tf \cdot idf$ ranking scheme from Salton and Buckley [Salton and Buckley (1988)] was developed for the vector space model as a way of weighting the relevance of a term to a document. The term frequency in the given document shows how important the term is in this document. The document frequency of the term shows how generally important the term is. A high weight in a $tf \cdot idf$ ranking scheme is reached by a high term frequency in the given document and a low document frequency of the term in the whole collection. Later, Robertson refined this scheme and covered it as an extension of the BIR model.

One of the most widely used and successful approaches to adhoc retrieval is the BM25 weighting scheme [Robertson et al. (1998)]:

$$w_{BM25}(t_i, d) = \frac{tf(t_i, d)}{tf(t_i, d) + 0.5 + 1.5 \cdot \frac{dl(d)}{avgdl}} \cdot \frac{\log \frac{N+0.5}{df(t_i)}}{\log N + 0.5} \tag{2.6}$$

Here tf is the term frequency, dl is the document length, df denotes the number of documents containing the term t_i, and N stands for the total number of documents in the collection.

2.2 Uncertain Inference models

This type of models can be seen a blend between logic and probability theory. An important aspect of this class of models is their extensibility and collection independence. In inference based models it is easy to combine different information sources: evidence is not limited to the query formulation, but can also include knowledge about the user, the domain etc. These parameters are collection-independent, whereas the relevance-based models contain parameters which have to be adjusted for every new collection.

2.2.1 Rijsbergen's model

In 1986 van Rijsbergen proposed the use of a non-classical conditional logic for IR [van Rijsbergen (1986)]. This would enable the evaluation of $P(d \rightarrow q)$ using the following logical uncertainty principle:

”Given any two sentences x and y; a measure of the uncertainty of $y \rightarrow x$ related to a given data set is determined by the minimal extent to which we have to add information to the data set, to establish the truth of $y \rightarrow x$.”

Van Rijsbergen's major contribution is the definition $P(d \rightarrow q)$ as the conditional probability $P(q|d)$.

This principle was the first attempt to make an explicit connection between non-classical logics and IR uncertainty modeling. However, when proposing the above principle, van Rijsbergen was not specific about which logic and which uncertainty theory to use. As a consequence, various logics and uncertainty theories have been proposed and investigated. The choice of the appropriate logic and uncertainty mechanisms has been a main research theme in logical IR modeling leading to a number of different approaches over the years.

2.2.2 Probabilistic Datalog

Datalog [Ullman (1988)] is a variant of predicate logic based on function-free Horn clauses which forms the theoretical basis for deductive databases. Probabilistic Datalog (pDatalog) [Fuhr (2000)] is a Datalog variant combined with probability theory, where every fact or rule has attached a probabilistic weight α, prefixed to the fact or rule:

0.5 female(X) :- person(X).
0.8 person(su).

A weight $\alpha = 1$ can be omitted. In that case the rule is called deterministic. Probabilistic Datalog is a powerful tool to model advanced retrieval functions and frameworks which are also capable of integrating additional knowledge.

HySpirit [Fuhr and Rölleke (1998)] is a retrieval engine for hypermedia retrieval integrating concepts from information retrieval (IR) and deductive databases by using probabilistic Datalog.

This basic approach was extended to four valued probabilistic Datalog [Rölleke (1998)], which also is the basis for the definition of POOL (Probabilistic Object-Oriented Logical Representation and Retrieval of Complex Objects).

A similar approach is POLAR [Frommholz (2008)] (Probabilistic Object-oriented Logics for Annotation-based Retrieval), a system to model structured annotation hypertexts, query them and perform annotation-based IR by applying probabilistic inference and implication.

PIRE [Nottelmann (2005)] uses Probabilistic Datalog++ (pDatalog++) for relational databases, which is an extension to pDatalog with aggregation, arbitrary functions for computing the probability of derived facts.

2.2.3 Inference network-based retrieval model

Inference networks are in fact Bayesian networks. As a probabilistic formalism for inference networks with uncertainty, Baysian inference networks have been described in [Pearl (1988)]. [Turtle and Croft (1990)] applied this formalism to document retrieval. A Bayesian network is usually depicted as a directed acyclic graph where nodes represent random variables and edges denote causal relationships. An inference network for IR consists of two layers, the document layer (which is built off line) and the query layer which is built on-line and can be interactively modified by the user [Kraaij (2004)]. The INQUERY system [Broglio et al. (1995)], based on the inference network-based retrieval model, has performed well on the TREC evaluation tasks. A nice property of the inference net framework is that multiple representations (e.g., single terms, phrases, controlled terms) of the same document can be represented in the same network, and also that an information need can be modeled by a parallel evaluation of different queries [Turtle and Croft (1990)].

2.3 Summary

Over the decades, many different retrieval models have been proposed. However, interestingly, empirical studies indicate that effective models seem to all boil down to some form of implementation of the three major retrieval heuristics (i.e., tf, idf, and document length normalization), and when optimized, these different models tend all to perform similarly well. The development of language modeling approaches has added another effective retrieval functions to the known $tf \cdot idf$ retrieval functions. In the next chapter, we will introduce this approach.

3

Language Models and Smoothing Methods

Language modeling (LM) approaches in information retrieval have mostly been developed in the last ten years. Through this short period, they have shown great promise for multiple retrieval tasks with very good empirical performance. Here we give an overview of the research in the area of language modeling for information retrieval.

A language model (LM) is a probabilistic mechanism for generating text. The first statistical language modeler was used by Claude Shannon for exploring the application of his newly founded theory of information to human language.

Language models are used in areas of human language technology, where they have played a central role in speech recognition research and system development [Jelinek (1998)].

Numerous other areas of textual language processing have also experimented with the language modeling approach.

Language model in IR uses techniques for estimation of the sampling probabilities to avoid the use of heuristics and rank documents by the probability that the document and the query are based on the same language model.

The language modeling approach to information retrieval was not proposed until the late 90's; this approach differs from traditional probabilistic approaches and was found to be an effective and theoretically attractive probabilistic framework for building IR systems.

[Ponte and Croft (1998b)] saw that the theoretical results and practical techniques resulting from the work with language models in human language technology are relevant to information retrieval. In the following years many other researchers made important contributions to this area. In [Ponte and Croft (1998b)], a language model is estimated for each document, and the operational procedure for ranking is to order documents by probability assigned to the input query text according to each document's model.

3.1 Language models history

A statistical language model, or more simply a language model, is a probabilistic mechanism for generating text. Such a definition is general enough to include an endless variety of schemes. However, a distinction should be made between generative models, which can in principle be used to synthesize artificial text, and discriminative techniques to classify text into predefined categories. The first statistical language modeler was Claude Shannon. In exploring the application of his newly founded theory of information to human language, Shannon considered language as a statistical source, and measured how well simple $n-$gram models predicted or, equivalently, compressed natural text. To do this, he estimated the entropy of English through experiments with human subjects, and also estimated the cross-entropy of the $n-$gram models on natural text [Shannon (1951)]. The ability of language models to be quantitatively evaluated in this way is one of their important virtues. Of course, estimating the true entropy of language is an elusive goal, aiming at many moving targets, since language is so varied and evolves so quickly.

For more than twenty-five years, language models have played a central role in speech recognition research and system development [Jelinek (1998)]. Based on a model of how a given language, such as English, is produced, a speech recognition system is able to choose among competing hypotheses of what was spoken. The language model assigns a probability to each utterance that can occur in English discourse. The recognizer then compares the probabilities assigned to each hypothesis in conjunction with other factors, and chooses the transcription hypothesis most likely to correspond to the speech signal being analyzed. Language models have also played a central role in statistical machine translation [Brown et al. (1990)]. As with speech recognition, a previously trained language model is a component of the decision procedure invoked to choose among competing hypotheses.

Numerous other areas of textual language processing have also experimented with the language modeling approach, especially with the application of hidden Markov models [Rabiner et al. (1983)]. This includes part-of-speech tagging [Cutting et al. (1992)], named-entity identification [Burger et al. (1998)], topic segmentation [Greiff et al. (2001), Yamron et al. (1998)], and selectional preference [Abney and Light (1999)].

An important factor in all technologies that use language modeling is that the model is learned. Statistical characteristics of the language of interest are extracted from a training corpus. From these statistics, parameters of the language model are estimated. The non trivial problem of parameter estimation is a major focus of all work on language modeling.

3.2 The Language Modeling Approach in the context of information retrieval research

3.2.1 Introducing Language Models in IR

The language model approach differs from traditional probabilistic approaches in interesting and subtle ways, and is fundamentally different from vector space methods. It is striking that the language modeling approach to information retrieval was not proposed until the late 1990s; however, until recently the information retrieval and language modeling research communities were somewhat isolated. The communities are now beginning to work more closely together, and research at a number of sites has confirmed that the language modeling approach is an effective and theoretically attractive probabilistic framework for building IR systems.

Language modeling approaches examine the generative process by which a user of an IR system creates a query. There are many different realization of this approach. Ponte and Croft [Ponte and Croft (1998b)] were one of the first to introduce the idea of applying statistical language modeling techniques to IR. Miller et al. [Miller et al. (1999)] also model the same process of query generation by using hidden Markov models.

In [Hiemstra (1998)] Hiemstra presented the linguistically motivated probabilistic model of information retrieval. Through estimation by linear interpolation, which is often used in the field of statistical natural language processing, he was able to present a probabilistic interpretation of $tf \cdot idf$ term weighting. He used the Cranfield collection and showed that the system based on the derived model performs better than the system based on a standard vector space model using classical $tf \cdot idf$ weights and cosine document length normalization.

Berger and Lafferty [Berger and Lafferty (1999)] formalized the process of a user generating a query as translation. The maximum likelihood ratio method by Ng [Spoken (1999)] starts with a different motivation, but uses a query generation probability in its ranking function.

3.2.2 Basic concepts

The basic idea of the language modeling approach is that the user who has an information need has some ideal document in his mind that he is seeking for. A query is something that the user formulates in an attempt to describe that ideal document. Therefore, with an appropriate model for query generation given, by postulating each document in the collection as a hypothetical ideal document, we can evaluate the likelihood that the query is the result of describing the particular document as an ideal document.

Relevance of a document to a query here is a likelihood that the user has that particular document in mind in the query generation process.

As illustrated by Berger and Lafferty [Berger and Lafferty (1999)], this process of document retrieval can be formalized by the noisy channel model [Cover and Thomas (1991)].

The encoding phase is where the user conceptualizes his information need into an ideal document. The process of the user describing this ideal document is taken to be a noisy channel. The task of the retrieval system is decoding the resulting query and figuring out the original information source, the ideal document posed by the user in his mind.

Ponte and Croft used the language modeling argument that an ideal document assumed by the user in his mind is the generative source for query formulation, and directly used the query generation probability for ranking documents.

For Miller et al, the query generation process is a Hidden Markov Model (HMM) process.

Berger and Lafferty formalized the entire retrieval process with a noisy channel model as just described, and modeled query formulation as statistical translation.

Despite all these variations in specific modeling of the retrieval process, the language modeling approach brings a theoretically sound, novel perspective into document retrieval. On the one hand, the concept of relevance is highly ambiguous and very hard to define. Attempting to probabilistically model relevance without explicitly and extensively modeling wide aspects of semantics may not be theoretically well justified. On the other hand, the idea of an IR user describing an ideal document through a query is easily understandable, and corresponds with the actual retrieval process very well. Furthermore, the approach allows us to take advantage of readily available language modeling techniques which have been extensively investigated by many researchers for over two decades in other areas of Natural Language processing (NLP) such as speech recognition, and have been shown to be very successful.

A statistical language model is a probability distribution over word sequences. It gives any sequence of words a potentially different probability. Given a language model, we can sample word sequences according to the distribution to obtain a text sample, where we may use such a model to "generate" text. Thus, a language model is also often called a generative model for text. A language model is very useful, where it allows us to answer many interesting questions related to information retrieval. For example, a language model may help answering the question: how likely would a user use a query containing the word "football" if the user wants to find information about sports?

If we enumerate all the possible sequences of words and give a probability to each sequence, the model would be too complex to estimate because the number of parameters is potentially infinite since we have a potentially infinite number of word sequences.

That is, we would never have enough data to estimate these parameters. Thus we always have to make assumptions to simplify the model. The simplest language model is the unigram language model in which we assume that a word sequence is generated by each word independently. Thus, the probability of a sequence of words would be equal to the product of the probability of each word.

Let V be the set of words in the vocabulary, and $t_1 ... t_n$ a word sequence, where $t_i \in V$ is a word :

$$P(t_1 ... t_n) = \prod_{i=1}^{n} P(t_i) \tag{3.1}$$

for $i = 1, ...|V|$.

It is easy to see that given a unigram language model θ, we have as many parameters as there are words in the vocabulary, i.e., $P(t_i|\theta)$, and they satisfy the constraint $\sum_{i=1}^{|V|} P(t_i|\theta) = 1$. Such a model essentially specifies a multinomial distribution over all the words. Clearly, given a language model θ, in general, the probabilities of generating two different documents d_1 and d_2, would be different, i.e., $P(d_1|\theta) \neq P(d_2|\theta)$. What kind of documents would have higher probabilities? Intuitively it would be those documents that contain many occurrences of the high probability words according to θ. In this sense, the high probability words of θ can indicate the topic captured by θ.

Now suppose that we have observed a document d (e.g., a short abstract of a text mining paper) which is assumed to be generated using a unigram language model θ, and we would like to infer the θ (i.e., estimate the probability of each word $t, P(t|\theta)$) based on the observed d. This is a standard problem in statistics (i.e., parameter estimation) and can be solved using many different methods.

One popular method is the maximum likelihood (ML) estimator, which seeks a model $\hat{\theta}$ that would give the observed data the highest likelihood (i.e., best explain the data) and it is easy to show that ML estimation of a unigram language model gives each word a probability equal to its relative frequency in d. That is,

$$P(t|\hat{\theta}) = \frac{tf(t,d)}{|d|} \tag{3.2}$$

If $tf(t,d)$ is the count of word t in d and $|d|$ is the length of d, or total number of words in d, the log-likelihood function can be written as follows:

$$\log P(d|\theta) = \sum_{t \in V} tf(t,d) \log P(t|\theta) \tag{3.3}$$

Because the ML estimate attempts to fit the data as much as possible it would give any word not seen in d a zero probability, which may not be reasonable, especially if d is a small sample. It is unreasonable according to our prior belief of what a word distribution characterizing the topic of a document should be like, intuitively. Had the author decided to write a longer document d, we would probably have been able to observe some of those unseen words. Adjusting the ML estimate to avoid zero probability is often called "smoothing". There are many different methods for smoothing a unigram language model; we will discuss some of them in section 3.4.

Although unigram language models are simple, they clearly make unrealistic assumptions about word occurrences in text. For example, if an author has started using a word in writing an article, the author would tend to use the same word again, which means that the probability of seeing a related word such as "program" would be much higher than if we had not seen "software" in the document. More sophisticated language models have thus been developed to address the limitations of unigram language models, see 6.3. For example, an n-gram language model would capture some limited dependency between words and assume the occurrence of a word depends on the proceeding $n - 1$ words. As a specific example, a bigram language model is defined as follows:

$$P(t_1...t_n) = P(t_1) \prod_{i=2}^{n} P(t_i|t_{i-1}) \tag{3.4}$$

Such a bigram language model can capture any potential local dependency between two adjacent words. There are also language models capturing remote dependencies through "triggers" [Rosenfeld (2000)]. A highly sophisticated language model is defined through a probabilistic context-free grammar; such a language model is explicitly structured based on the grammar of a language [Manning and Schütze (1999)].

While theoretically speaking, we would like to adopt a sophisticated language model that can model our language more accurately; in reality, we often have to take a tradeoff. This is because as the complexity of a language model increases, so does the number of parameters. As a result, we would need much more data to estimate the parameters. With limited amount of data, our estimate of parameters would not be accurate. The computational cost of complex language models is also a concern for all large-scale retrieval applications. So far, the simplest unigram language model has been shown to be quite effective for information retrieval, while more complex language models such as bigram or trigram language models tend to improve not much over the unigram language model. One reason may be the problem of data sparseness, which makes the estimated complex language model inaccurate. From a retrieval perspective, the non-promising performance of complex language models may also be related to non-optimal weighting of bigrams and trigrams; indeed, when they are combined with unigrams, we must avoid over-rewarding matching multiple words in a phrase [Mitra et al. (1997)]. Another reason why unigram language models seem to perform very well is because the retrieval task is a (relatively) easy task compared with some other language understanding tasks such as machine translation. Information about word presence or absence and word frequencies may be sufficient to determine the relevance of a document while the exact word order may not be so important (as, e.g., in the case of machine translation).

For machine translation, unigram language models are clearly insufficient and more sophisticated language models would be needed [Brown et al. (1990)]. Also, for speech recognition, modeling word order is obviously very important [Jelinek (1998)].

In information retrieval, what we care about is how effective a language model is for retrieval. Thus, we would use a language model for ranking documents and evaluate the accuracy of ranking. That is, we would evaluate a language model based on its contribution to retrieval accuracy. This is an indirect way of evaluating the quality of a language model because we assess a language model together with other components of a retrieval system, and the retrieval performance we see can be potentially affected by many other factors, not just the language model. Since a language model is a probabilistic model of text data, a more direct way of evaluating a language model would be to assess how well the model fits the data to be modeled (i.e., test data). For example, we may compute the likelihood of the test data given a model to be evaluated, a higher likelihood would indicate a better fit, thus a better language model. Note that the relative performance of two language models may be different when using these two different evaluation strategies. This is because there may be some gap between the task and the language model. As a result, fitting the data well does not always imply better performance for the task. A main goal in research on using language models for retrieval is to design appropriate retrieval models so that an improved language model (improved in terms of direct evaluation) would also lead to improved performance for the retrieval task. If such a correlation exists, we would have some guidance on how to find a better retrieval model. We may find a better retrieval model through improving language models and/or estimation of language models.

3.3 Simple Query Likelihood Retrieval Model

The basic idea of the simple query likelihood retrieval model is to estimate a language model for the document, and then compute the likelihood of the query according to the estimated language model. The documents are then ranked based on their query likelihood scores. If we have a query q and a document d, then θ_d is the language model estimated on document d and the score of document d w.r.t. query q is defined as the conditional probability $P(q|\theta_d)$. θ_d should be interpreted as modeling the queries that a user would use in order to retrieve document d. Although the parameter θ_d in the query likelihood scoring formula is often called a document language model, it is really a model for queries, not documents. For a given query q, the query likelihood retrieval model would test each document d to see whether a user would likely use the current query q to retrieve d if the user likes document d, and rank the documents based on this query likelihood. If a user who likes document d would always use query q to retrieve d, we would have $P(q|\theta_d) = 1$. Thus, if we see query q again, we would rank d on the top because it has the highest query likelihood. On the other hand, if a user who wants to retrieve document d would never use query q, we would have $P(q|\theta_d) = 0$. Thus, we would rank this document at the bottom for query q. As a general retrieval model, the query likelihood retrieval model must have some way to score any document with respect to any query. The solution to this problem taken by the simple query likelihood retrieval model is to simply estimate θ_d based on d, i.e., we would use document d to approximate the queries that a user would use to retrieve D. Several models of θ_d will be presented in the following sections.

3.3.1 Multinomial θ_d

In a multinomial θ_d, a sequence of words will be generated by generating each word independently. Thus, a multinomial model θ_d would have the same number of parameters (i.e., word probabilities) as the number of words in the vocabulary set V, and the query likelihood would be

$$P(q|\theta_d) = \prod_{t \in V} P(t|\theta_d)^{tf(t,q)} \tag{3.5}$$

where $tf(t,q)$ is the count of word t in query q. With such a model, the retrieval problem is reduced to the problem of estimating $P(t|\theta_d)$.

3.3.2 Multiple Bernoulli θ_d

In the multiple Bernoulli θ_d a binary random variable $X_i \in 0,1$ is defined for each word t_i to indicate whether t_i is present ($X_i = 1$) or absent ($X_i = 0$) in the query. Let us assume that the presence of each word is independent of each other. Thus, a multiple Bernoulli model θ_d would again have the same number of parameters as the number of words in the vocabulary. Such a model can consider presence and absence of words in the query. According to the multiple Bernoulli model, the query likelihood would be

$$P(q|\theta_d) = \prod_{t_i \in q^T} P(X_i = 1|\theta_d) \prod_{t_j \notin q^T} (1 - P(X_j = 1|\theta_d)) \tag{3.6}$$

where the first product is for words in the query, and the second for words not occurring in the query and the retrieval problem has been reduced to the problem of estimating $P(X_i = 1|\theta_d)$.

3.3.3 Multiple Poisson θ_d

According to the Poisson distribution, the probability of observing x counts of a word during time period t from a Poisson process with parameter λ is

$$P(X = x|\lambda) = \frac{\mathrm{e}^{-\lambda t}(\lambda t)^x}{x!} \tag{3.7}$$

To model the counts of a word in the query, the query length m is taken as the length of the time period. Thus, the query likelihood is:

$$P(q|\theta_d) = \prod_{t_i \in V} \frac{\mathrm{e}^{-\lambda_i m}(\lambda_i m)^{tf(t_i,q)}}{tf(t_i,q)!} \tag{3.8}$$

and the retrieval problem has been reduced to the problem of estimating the λ_i.

3.3.4 Comparison of the three models

Most research so far has focused on the multinomial model, even though multiple Bernoulli was the model used in the pioneering work by Ponte and Croft [Ponte and Croft (1998b)]. When each word in a document is regarded as a sample of a multiple Bernoulli process where only this word occurred, but all other words did not, it can be shown that the estimation of multiple Bernoulli is related to the estimation of the multinomial model [Metzler et al. (2004)]. Empirically, there has been some evidence that multinomial outperforms multiple Bernoulli [Song and Croft (1999)], but a more systematic comparison between them is needed in order to draw definitive conclusions. The Poisson model appears to have some advantages [Mei et al. (2007)], but there has not been much work on this model yet.

3.3.5 Basic multinomial model

Here we present the basic multinomial language model, which forms the basis for Zhai and Lafferty's model.

Let q denote a query containing the set of terms q^T, and d is a document with the set of terms d^T. Furthermore, let t_i denote a term and C stands for the collection. Then we compute the conditional probability of observing the query q given the document d as follows:

$$\begin{aligned} P(q|d) &= \prod_{t_i \in q^T} P(t_i|d) \\ &= \prod_{t_i \in q^T \cap d^T} P_s(t_i|d) \prod_{t_i \in q^T - d^T} P_u(t_i|d) \\ &= \prod_{t_i \in q^T \cap d^T} \frac{P_s(t_i|d)}{P_u(t_i|d)} \prod_{t_i \in q^T} P_u(t_i|d) \end{aligned} \tag{3.9}$$

Here we have the following probabilities:

$P(d)$ Probability that d generates an arbitrary query.

$P_s(t_i|d)$ Probability that d generates term t_i, given that t_i occurs in d.

$P_u(t_i|d)$ Probability that d generates term t_i, given that t_i does not occur in d.

Where s means "seen", u stands for "unseen" and the probability of an unseen word is typically taken as being proportional to the general frequency of the word, and computed using the document collection, i.e. the collection language model $P(t_i|C)$, which can be estimated as $P_{avg}(t_i|C) = \frac{cf(t_i)}{cs}$, where $cf(t_i)$ is the raw count of term t in the collection and cs is the raw collection size or the total number of tokens in the collection.

3.3.6 Zhai/Lafferty model

Building on the basic model (3.9) the core idea of Zhai and Lafferty model [Zhai and Lafferty (2001b)] is the estimation of the probability $P_u(t_i|d)$ of terms not occurring in the document, by means of the following formula:

$$P_u(t_i|d) = a_d P_{avg}(t_i|C) \tag{3.10}$$

Here a_d is a document-dependent constant estimated in the following way:

$$a_d = \frac{1 - \sum_{t_i \in q^T \cap d^T} P_s(t_i|d)}{1 - \sum_{t_i \in q^T \cap d^T} P_{avg}(t_i|C)} \tag{3.11}$$

Regarding the logarithmic form of (3.9), their retrieval function yields:

$$\begin{aligned} \log P(q|d) = & \sum_{t_i \in q^T \cap d^T} \log \frac{P_s(t_i|d)}{a_d \cdot P_{avg}(t_i|C)} + n \log a_d \\ & + \sum_{t_i \in q^T} \log P_{avg}(t_i|C) \end{aligned} \tag{3.12}$$

where n is the length of the query.

The core problem in language model estimation is smoothing, which adjusts the maximum likelihood estimator so as to correct the inaccuracy due to data sparseness.

3.4 Smoothing methods

A straight forward method for estimating the parameters $P(t_i|d)$ is

$$P_{ml}(t_i|d) = \frac{tf(t_i, d)}{|d|} \tag{3.13}$$

One problem with this maximum likelihood estimator is that an unseen word in document d would get a zero probability, making all queries containing an unseen word have zero probability $P(q|d)$. To solve this problem we need to smooth the result of this estimator so that we do not assign zero probability to unseen words and can improve the accuracy of the estimated language model in general. We set the probability of an unseen word to the average probability of that word in the whole collection. This is to say that if we do not observe a word in the document, we would assume that the probability of such a word is the same as the probability of seeing the word in any document in the whole collection. This ensures that none of the words in the collection will get a zero probability.

The most popular smoothing methods will be presented in the following.

3.4.1 The Good-Turing Estimate

The Good-Turing estimate [Manning and Schütze (1999)] has been known to be a very reliable method for deriving smoothed term frequencies from limited-sized observed data such as a single document. The smoothed, or "modified", term frequency of a term t with a raw term frequency tf is given by the Good-Turing estimate as follows:

$$tf^* = (tf + 1)\frac{E(N_{tf+1})}{E(N_{tf})} \tag{3.14}$$

N_{tf} is a number of terms whose term frequency is tf, and $E(N_{tf})$ is an expected value of such a number. In practice, it is not realistically possible to compute $E(N_{tf})$, so usually the observed value is substituted instead. This can be a problem when N_{tf+1} happens to be 0, and especially with the highest frequency term, N_{tf+1} is always 0. Therefore, curve-fitting or any other pre-smoothing is typically done on raw term frequencies before the Good-Turing estimate is used.

3.4.2 The Jelinek-Mercer method

The Jelinek-Mercer method [Jelinek and Mercer (1980)] involves a linear interpolation of the maximum likelihood model with the collection model, using a smoothing coefficient λ to control the influence of the collection model. The resulting probability estimate is called $P_{s,\lambda}$ here:

$$P_{s,\lambda}(t_i|d) = (1-\lambda) \cdot P_{ml}(t_i|d) + \lambda \cdot P_{avg}(t_i|C) \tag{3.15}$$

3.4.3 Bayesian parameter estimation

Typical smoothing methods in language models are length-independent. On the other hand, it is obvious, that the maximum likelihood estimate is more biased for shorter documents. When the documents in the collection are of almost uniform length (which e.g. is the case for the largest part of the TREC collections), this effect can be compensated by document-independent smoothing parameters. However, in a collection with a big variation in document lengths, a document-dependent smoothing factor may be more adequate. One possible approach following this strategy is Bayesian parameter estimation. Since a language model is a multinomial distribution, the corresponding conjugate prior is the Dirichlet distribution with parameters

$$(\mu P(t_1|C), \mu P(t_2|C), ..., \mu P(t_n|C)) \tag{3.16}$$

and the estimate of $P_u(t_i, d)$ is given as

$$P_{s,\mu}(t_i|d) = \frac{tf(t_i, d) + \mu P_{avg}(t_i|C)}{\sum_{t_i \in d^T} tf(t_i, d) + \mu} \tag{3.17}$$

where $tf(t_i, d)$ is the number of occurrences of t_i in d, μ is a parameter for adjusting the amount of smoothing applied. The optimal prior μ seems to vary from collection to collection, though in most cases, it is around 2,000.

3.4.4 Absolute discount

The following method is similar to Jelinek-Mercer, but differs in that it discounts the seen word probability by subtracting a constant instead of multiplying it by $(1 - \lambda)$.

So the estimate of $P_u(t_i, d)$ is given as

$$P_{s,\delta}(t_i|d) = \frac{\max(tf(t_i, d) - \delta, 0)}{\sum_{t_i \in d^T} tf(t_i, d)} + \sigma P(t_i|C) \tag{3.18}$$

where

δ is a discounting constant,

$\sigma = \frac{\delta \cdot |d^T|}{|d|}$, with

$|d|$ denoting the document length, and

$|d^T|$ is the number of unique terms in document d

Many smoothing methods were applied to language models for information retrieval (e.g. [Miller et al. (1999)], [Hiemstra (1998))]. Other smoothing techniques have been evaluated e.g. in [Lafferty and Zhai (2001a)].

3.4.5 Two-Stage Smoothing

The two-stage smoothing strategy [Zhai and Lafferty (2002)] explicitly captures the different influences of the query and the document collection on the optimal settings of smoothing parameters. In the first stage, the document language model is smoothed using a Dirichlet prior with the collection model as the reference model. In the second stage, the smoothed document language model is further interpolated with a query background model.

$$P_{\lambda,\mu}(t|\theta_d) = (1-\lambda)\frac{tf(t,d) + \mu P_{avg}(t|C)}{|d| + \mu} + \lambda P(t|U) \tag{3.19}$$

The query background model $P(t|U)$ is in general different from the collection language model $P(t|C)$. With insufficient data to estimate $P(t|U)$, however, they assume that $P(t|C)$ would be a reasonable approximation of $P(t|U)$. In practice, the performance is usually not much better than a well-tuned single stage smoothing method such as Dirichlet prior.

3.4.6 Dual Role of Language Model Smoothing for IR

Combining the estimate from a single document with the estimate from the whole document collection in various manners is a technique that is very commonly used in statistical language modeling. Zhai and Lafferty [Zhai and Lafferty (2001a)] hypothesized that this type of smoothing plays a dual role, which is:

1. It improves the reliability of the model, especially by assigning non-zero probabilities to terms that do not occur in the document.
2. It facilitates the generation of terms in the query that are commonly used in general or are particularly typical in the collection.

The original motivation for smoothing is its first role of solving the problem of zero probability, but Zhai and Lafferty argue that as a query becomes longer and more verbose, or in more descriptive, the relative weight of the second role that smoothing plays becomes more significant.

The use of smoothing to facilitate the generation of common words can be clearly seen in both of the Hidden Markov retrieval Model (HMM) and the retrieval as statistical translation model. The simplest HMM retrieval model consists of two stages, one of which corresponds to the term generation based on the general English model. In the statistical translation model, the term $t(.| < null >)$ provides the background model that can account for common expressions such as "I want documents that discuss ..." The role that those HMM state for general English generation and null word translation play, other than to smooth probability distributions, is to allow the query to not just consist of only keywords but also be descriptive with words from a common vocabulary of the domain.

3.5 Cross-Entropy

Cross-Entropy is a ranking measure inspired by the recent successes of language modeling. Let $P(t|R)$ denote the language model of the relevant class and for every document d, let $P(t|d)$ denote the corresponding document language model.

Cross-entropy is a natural measure of divergence between two language models:

$$H(R||d) = -\sum_{t \in T} P(t|R) \log P(t|d) \tag{3.20}$$

Documents are ranked by increasing cross-entropy and the terms with $P(t|R) = 0$ are excluded. This model is a general one, since it allows to estimate $P(t|R)$ in any fashion, where there is also a special case when we estimate $P(t|R)$ as the relative frequency of the terms t in the user query q.

Lafferty and Zhai proposed a document ranking method based on a risk minimization framework [Lafferty and Zhai (2001b)] where they suggest to use the relative entropy of Kullback-Leibler (KL) divergence between a distribution representing the query and a distribution for the document.

The KL divergence is either zero when the probability distributions are identical or has a positive value, quantifying the difference between the distributions by the number of bits which are wasted by events from the distribution P with a "code" based on distribution Q.

However, KL has some less attractive characteristics; it is not symmetric and does not satisfy the triangle inequality and thus is not a metric.

3.6 Cross-lingual information retrieval (CLIR)

CLIR is the task of retrieving documents in one language (e.g., English) with a query in another language (e.g., Chinese). A major challenge in cross-lingual IR is to cross the language barrier in some way, typically involving translating either the query or the document from one language to the other. [Lavrenko et al. (2002)] applied language models to CLIR, one of their methods is to leverage a bilingual dictionary to induce a translation model $P(t^S|t^T)$ and to use this translation model to convert the document language model $P(t^T|d)$, which is in the target language, to a document language model for the source language $P(t^S|d)$. That is,

$$\begin{aligned} P(t^T|\theta_q) &= \sum_{\theta_d \in \Theta} P(\theta_d) P(t^T|\theta_d) \prod_{i=1}^{m} P(w_i|\theta_d) && (3.21) \\ &= \sum_{\theta_d \in \Theta} P(\theta_d) P(t^T|\theta_d) \prod_{i=1}^{m} \sum_{t \in V^T} P(w_i|t) P(t|\theta_d) && (3.22) \end{aligned}$$

Here V^T is the vocabulary set of the target language (i.e, the language of the document d), $P(w_i|t)$ is the translation model, i.e. the probability of "translating" word t into w_i. This translation model can be understood by imagining a user who likes document d would formulate a query in two steps. In the first, the user would sample a word from document d; in the second, the user would "translate" the word into possibly another, different but semantically related word. These models have been shown to achieve very good retrieval performance (90%-95% of a strong monolingual baseline).

3.7 Comparisons with traditional probabilistic IR approaches

The language modeling approach has introduced a new family of probabilistic models to IR. Several researchers have attempted to relate this new approach to the traditional probabilistic IR approaches and compare their difference. [Sparck Jones and Robertson (2001)] examine the notion of relevance in the traditional probabilistic approach (PM) and the new language modeling approach (LM), and point out that two distinctions should be made between the two approaches.

The first and what they call surface distinction is that while in both approaches, a good match on index keys between a document and a query implies relevance, relevance figures explicitly in PM but is never mentioned in LM.

The second and what they find the more important difference is that the underlying principle of LM is to identify the ideal document that generates the query rather than a list of relevant documents. Thus once this ideal document is recovered, retrieval stops.

Because of this, they argue that it is difficult to describe important processes such as relevance feedback in the existing LM approaches. [Lafferty and Zhai (2001b)], [Lafferty and Zhai (2001a)] and [Lavrenko and Croft (2001)] address these issues directly and suggest new forms of the LM approach to retrieval that are more closely related to the traditional probabilistic approach by [Robertson and Sparck Jones (1976)]:

A document could be thought of as being generated from a query using a binary latent variable that indicates whether or not the document is relevant to the query.

They show through mathematical derivations that, if a similar binary latent variable is introduced to LM, these two methods are on equal footing in terms of the relevance ranking principle and interpretation of the ranking process. However, this does not mean that PM and LM are just a reversion of each other.

The difference goes beyond a simple application of Bayes' law. They point out that document length normalization is a critical issue in PM but it is not in LM. Another difference is that in LM we have more data for estimating a statistical model than in PM which is the advantage of "turning the problem around".

Both the risk-minimization framework suggested by [Lafferty and Zhai (2001b)], [Lafferty and Zhai (2001a)] and the relevance model suggested by [Lavrenko and Croft (2001)] move away from estimating the probability of generating query text (the query-likelihood model) to estimating the probability of generating document text (document-likelihood) or comparing query and document language models directly. [Greiff (2001)] suggests that the main contributions of LM to IR lie in the recognition of the important role of parameter estimation in modeling and the treatment of term frequency as the manifestation of an underlying probability distribution rather than as the probability of word occurrence itself.

[Zhai and Lafferty (2002)] point out that traditional IR models rely heavily on ad hoc parameter tuning to achieve satisfactory performance whereas in LM, statistical estimation methods can be used to set parameters automatically. Hiemstra and de Vries [Hiemstra and de Vries (2000)] relate LM to traditional approaches by comparing Hiemstra's model [Hiemstra (1998)] with the $tf \cdot idf$ term weighting and the combination with relevance weighting as done in the BM25 algorithm. They conclude that LM and PM have important similarities in that LM provides a probabilistic interpretation and justification of the $tf \cdot idf$ weighting and gives insight in why the combination of it with relevance weighting in BM25 is effective.

Fuhr [Fuhr (2001a)] shows that the LM approach can be related to other probabilistic retrieval models like e.g. [Wong and Yao (1995)] in the framework of uncertain inference.

3.8 Document length and retrieval systems

Document length is widely recognized as an important factor for adjusting retrieval systems. Many models tend to favor the retrieval of either short or long documents and, thus, a length-based correction needs to be applied for avoiding any length bias. In language modeling for information retrieval, smoothing methods are applied to move probability mass from document terms to unseen words, which is often dependent upon document length. The problem of document length normalization [Singhal et al. (1996)], [Robertson and Walker (1994)] is ensuring that documents of particular lengths are not unduly favored over documents of other lengths by the retrieval model. The need to account for this problem is because [Singhal et al. (1996)]:

1. Long documents tend to have more occurrences of different terms which means that long documents are more likely to match query terms.
2. As the length of the document increases, the number of times a particular term occurs in the document also increases, which in turn increases the matching score.

Consequently, the term weights in a document need to be penalized in accordance with document length (and thus the document been normalized). Accounting for document length effects within a retrieval algorithm tends to improve performance [Singhal et al. (1996)], [Amati (2003)], [Chowdhury et al. (2002)]. Although these normalization issues have been extensively studied in the context of many IR models, such as the Vector Space Model [Singhal et al. (1996)], the classic probabilistic model [Robertson et al. (1995)] and the Divergence from Randomness Model [Amati and van Rijsbergen (2002a)], the effect of document length has scarcely been discussed in the context of Language Modeling.

In LM [Ponte and Croft (1998b)], [Jones et al. (2003)] and [Zhai and Lafferty (2001b)], smoothing methods are applied to move probability mass from document terms to unseen words when constructing a LM for a document. This provides an implicit length normalization component, where the amount of smoothing applied affects the distribution of the lengths in the retrieved set of documents. The smoothing method and parameter estimation will dictate whether longer or shorter documents are favored, or not.

We consider document length in our models (chapter 7) and our experiments show that the retrieval performance is highly affected as the document length varies and it is of major importance in language modeling for information retrieval. The hypothesis that the probability of relevance is correlated with document length was empirically supported.

4 XML retrieval

XML has grown into a huge topic, inspiring many technologies and branching into new areas. XML information retrieval differs from traditional IR in that elements rather than documents are units of retrieval, and that a varying granularity of answers can be returned in respose to a request. Here we present a very brief description about XML and XML retrieval.

4.1 XML

XML, the eXtensible Markup Language, is a W3C-endorsed standard for document markup. It defines a generic syntax used to mark up data with simple, human-readable tags. It provides a standard format for computer documents that is flexible enough to be customized for domains as diverse as web sites, electronic data interchange, vector graphics, genealogy, real estate listings, object serialization, remote procedure calls, voice mail systems, and more. XML is a metamarkup language for text documents. Data are included in XML documents as strings of text. The data are surrounded by text markup that describes the data.

The XML specification defines the exact syntax this markup must follow: how elements are delimited by tags, what a tag looks like, what names are acceptable for elements, where attributes are placed, and so forth. XML allows developers and writers to invent the elements they need as they need them. The X in XML stands for eXtensible, which means that the language can be extended and adapted to meet many different needs. The markup in an XML document describes the structure of the document. It shows which elements are associated with other elements. In a well-designed XML document, the markup also describes the document's semantics.

The markup says nothing about how the document should be displayed. That is, it does not say that an element is bold or italicized or a list item. XML is a structural and semantic markup language, not a presentation language. XSL (the eXtensible Stylessheet Language) is the preferred style sheet language of XML. With XSL one can add display information to a document. One way to use XSL is to transform XML into HTML before it is displayed by the browser.

The markup permitted in a particular XML application can be documented in a schema. Particular document instances can be compared to the schema. Documents that match the schema are said to be valid. That is, whether a document is valid or invalid depends on which schema you compare it to. Although XML is quite flexible in the elements it allows, it is quite strict in many other respects. The XML specification defines a grammar for XML documents that says where tags may be placed, what they must be like, which element names are legal, how attributes are attached to elements, and so forth. This grammar is specific enough to allow the development of XML parsers that can read any XML document. Documents that satisfy this grammar are said to be well-formed. XML processors reject documents that contain well-formedness errors. Not all documents need to be valid. For many purposes it is enough that the document is well-formed.

Some standards

There is a set of standards that is related to XML. Some of the better known ones are Document Object Model (DOM), Simple API for XML (SAX), eXtensible Style Sheet Language (XSL), XML Linking language (XLink), XML pointer Language (Xpointer), XQuery, Extensive Style Sheet Language Transformation (XSLT), XML path Language (XPath) etc.

DTD The purpose of a DTD (Document Type Definition) is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.

XML Schema XML Schema is an XML based alternative to DTD which describes the structure of an XML document. It is more complicated but allows for more precise specifications (e.g. w.r.t. data types) of XML documents.

XPath XPath (XML Path language) is a query language defined by the W3C [1]. Its primary purpose is to access or navigate to components of an XML document. In addition, XPath provides basic facilities for the manipulation of strings, numbers and Booleans.

XQuery XQuery is a powerful query language for finding and extracting (querying) data from XML documents and it is built on XPath expressions. XQuery for XML is like SQL for databases and it adds to XPath the possibility to query multiple documents and combine the results into new XML fragments (result construction). However it is mostly appropriate for data-centric XML retrieval. This is because its text search capabilities are limited and, in addition, it does not provide any ranking of results, the latter being crucial in content-oriented XML retrieval. These shortcomings led

[1] http://www.w3.org/

to the specification and development of XQuery/Full-text, which overcomes most of these limitations.

Next wave of the internet technology ?

XML allows customized tags to add more semantics to the web content page. Today's search engines are mostly based on word matching, rather than considering the semantics of both the query formulation and the document text. For example, a search on "Thinkpad" will give you all documents which contain the word "Thinkpad" when the user is more interested in product information pages on "Thinkpad". To solve this problem, one can use XML customized tags to classify data with semantics as "product", "related-product", etc. Another important feature of XML is interoperability support: different applications can communicate and extract information from the same XML document as long as they use the same DTD.

4.2 INEX

The INitiative for the Evaluation of XML retrieval (INEX) [Fuhr et al. (2003)], which was set up in 2002, established an infrastructure and provided means, in the form of large test collections and appropriate scoring methods, for evaluating how effective content-oriented XML search systems are [Lalmas and Tombros (2007)].

Within INEX, the aim of an XML retrieval system is "to exploit the logical structure of XML documents to determine the best document components, i.e. best XML elements, to return as answers to queries" [Blanken et al. (2003)]. Query languages have been developed in order to allow users to specify the nature of these best components. Indexing strategies have been developed to obtain a representation not only of the content of XML documents, but their structure. Ranking strategies have been developed to determine the best elements for a given query.

4.2.1 XML structure

According to [Fuhr and Lalmas (2007)], structure in XML retrieval may be regarded at the following levels:

1. **Nested Structure** "Whereas classical retrieval regards documents as atomic units, the XML markup of a document immediately implies a nested, tree-like structure."
2. **Named Fields** Consider only the tag names but not the nested structure.
3. **XPath** "XPath provides full expressiveness for navigating through the document tree, by parent/child and ancestor/descendant relationships, whereas horizontal navigation is supported via operators like following/preceding, following-sibling and preceding-sibling; in addition, there are the attribute and namespace axis."
4. **XQuery** XQuery offers an even higher expressiveness than XPath, by allowing for aggregation and restructuring of the found elements.

4.2.2 Types of XML IR queries in INEX

The XML query languages used in INEX, have been classified as content-only or content-and-structure query languages.

4.2.2.1 Content-only queries

Content-only queries make use of content constraints to express user information needs. In their simplest form, they are made of words, which have historically been used as the standard form of input in information retrieval. They are suitable for XML retrieval scenarios in which users do not know or are not concerned with the document structure when expressing their information needs. Although only the content aspect of the information need is being specified, XML retrieval systems must still determine what are the best fragments, i.e., the XML elements at the most appropriate level of granularity, to return as answers to a query.

4.2.2.2 Content-and-structure queries

Content-and-structure queries provide a means for users to specify their content and structural information needs. There are three main categories of content-and-structure query languages, namely tag-based languages, path-based languages, and clause-based languages, where the complexity and the expressiveness of these query languages increase from tag-based to clause-based queries. From a user perspective, this increase in expressiveness and complexity often means that content-and-structure queries are hard to write. Nonetheless, they can be very useful for expert users in specialized scenarios, such as patent retrieval and genomic search.

The following is an example from NEXI (Narrowed Extended XPath I) [Trotman and Sigurbjornsson (2005)] which was introduced in INEX 2004 as a query language for specifying both structured and unstructured queries on XML document.

```
//article[about.//abs|kwd), description logics)]//fm//au
```

The above query example asks for articles that have an element `//fm//au`, as well as `abs` elements(tags) which mention "description logics" or `kwd` elements that mention "description logics".

4.3 DBMS and XML

An XML document is a collection of data. In many ways, this makes it no different from any other file, after all, all files contain data of some sort. As a "database" format, XML has some advantages. For example, it is self-describing (the markup describes the structure and type names of the data, although not the semantics), it is portable

(Unicode), and it can describe data in tree or graph structures. It also has some disadvantages. For example, it is verbose and access to the data is slow due to parsing and text conversion.

XML provides many of the things found in databases: storage (XML documents), schemas (DTDs, XML Schemas), query languages (XQuery, XPath, XQL, etc.), programming interfaces (SAX, DOM, JDOM), and so on. On the other hand, it lacks many of things found in real databases; like e.g. efficient storage, indexes, security, transactions and integrity, multi-user access, triggers, queries across multiple documents.

XML documents share the same data type underlying the XML paradigm: ordered trees. Tree nodes represent document elements, attributes or text data, while edges represent the element-subelement (or parent-child) relationship. There are many approaches to numbering XML element nodes. As an example for the document from figure 4.1, we used tree traversal order, where a tree node is assigned a pair of (preorder,postorder) tree traversal numbers. Element u is an ancestor of element v iff $u.preorder < v.preorder$ and $v.postorder < u.postorder$. Viewing XML documents as trees is often convenient when one wants to describe structural properties of documents. For instance, the level of a node in the XML tree is its distance from the root node of the document (the level of the root node is 0). Similarly, we define the out-degree of an XML element, by the number of its children.

For the example document from figure 4.1, figure 4.2 shows how the encoded XML document can be represented as a tree inside a database system (the nodes are represented as circles and the text "pcdata" is represented as rectangles), i.e., the pre/post plane, is represented in relational tables. Two of our tables, the node table (4.1) and the weight table (4.2) show an example after indexing the INEX collection (5.6.2.3), where we have stored information about each element, like pre/post order, element name, type, level, out degree and the XPath. The information then extracted from the node table is exploited in our model. We are using XML in a relational database and not as database format.

We have upgraded our model, which will be presented in chapter 7, to deal with XML retrieval, taking advantage of post degree and out order parameters. See Chapter 8 for details.

```
<?xml version="1.0" encoding="UTF\-8 "?>
<book class="H.3.3">
<author>John Smith</author>
<title>XML Retrieval</title>
<chapter>
     <heading>Introduction</heading>
This text explains all about XML and IR.
 </chapter>
<chapter>
    <heading> XML Query Language XQL</heading>
           <section>
        <heading>Examples</heading>
           </section>
    <section>
          <heading>Syntax</heading>
           Now we describe the XQL syntax.
    </section>
</chapter>
</book>
```

Figure 4.1: Simple XML document

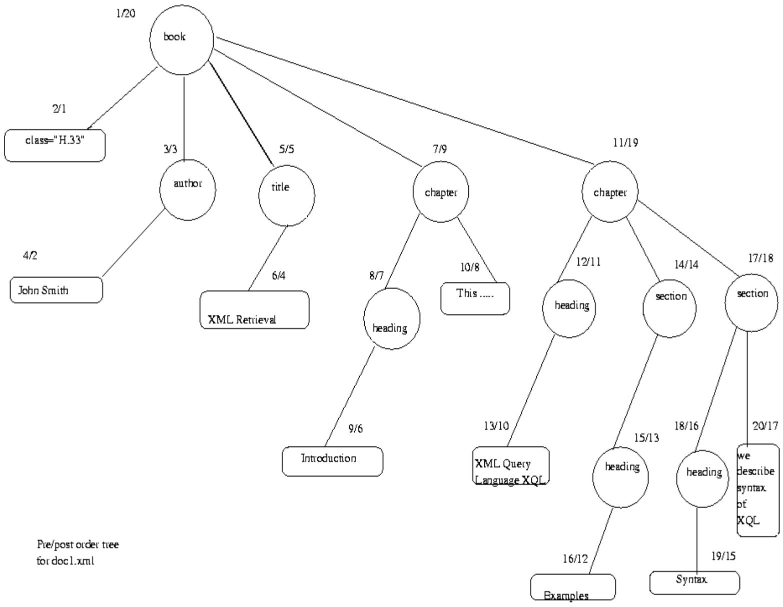

Figure 4.2: Pre/post scheme for the example document in fig 4.1

Table 4.1: Node table

doc	*Pre*	*Post*	*Element Name*	*Type*	*Level*	*Out degree*	*Xpath*
cg/2002/g5022	1	20	book	NODE	0	5	/book[1]
cg/2002/g5022	2	1	@class	ATT	1	0	/book[1]/class[1]
cg/2002/g5022	3	3	author	NODE	1	1	/book[1]/author[1]
cg/2002/g5022	4	2	PCDATA	TEXT	2	0	/book[1]/author[1]
cg/2002/g5022	5	5	title	NODE	2	1	/book[1]/title[1]
cg/2002/g5022	6	4	PCDATA	TEXT	2	0	/book[1]/title[1]
cg/2002/g5022	7	9	chapter	NODE	1	2	/book[1]/chapter[1]
cg/2002/g5022	8	7	heading	NODE	2	1	/book[1]/chapter[1]/heading[1]
cg/2002/g5022	9	6	PCDATA	TEXT	3	0	/book[1]/chapter[1]/heading[1]
cg/2002/g5022	10	8	PCDATA	TEXT	2	0	/book[1]/chapter[1]

Table 4.2: Weight table

doc	*Term*	*Prob*
cg/2002/g5022@1	class	0.002
cg/2002/g5022@1	john	0.0027
cg/2002/g5022@1	smith	0.0058
cg/2002/g5022@1	xml	0.0013
cg/2002/g5022@1	retriev	0.0015
cg/2002/g5022@1	query	0.0007
cg/2002/g5022@1	language	0.0005
cg/2002/g5022@1	describ	0.001
cg/2002/g5022@1	syntax	0.001
cg/2002/g5022@1	xql	0.0017

5

Relevance and Evaluation

Relevance has always been taken as fundamental to information retrieval. This chapter discusses the role of relevance and how to evaluate retrieval effectiveness, contents of the test collections like a document collection, a test suite of information needs expressible as queries and a set of relevance judgements.

5.1 Relevance and evaluation in information retrieval

How do we know which IR techniques are effective in which applications? Should we use stop lists? Should we stem? Should we use inverse document frequency weighting? Information retrieval has developed as a highly empirical discipline, requiring careful and thorough evaluation to demonstrate the superior performance of novel techniques on representative document collections. The question of which measures to use to evaluate retrieval effectiveness has received much attention in the literature. Different evaluation measures have different properties with respect to how closely correlated they are with user satisfaction criteria, how easy they are to interpret, how meaningful average values are, and how much power they have to discriminate among retrieval results.

The standard approach to information retrieval system evaluation revolves around the notion of relevant and non-relevant documents. With respect to a user information need, a document in the test collection is given a binary classification as being either relevant or non-relevant. This decision is referred to as the gold standard or ground truth judgement of relevance. The test document collection and suite of information needs to be of a reasonable size: you need to average performance over fairly large test sets, as results are highly variable over different documents and information needs.

A document is relevant if it addresses the stated information need, not because it just happens to contain all the words in the query. This distinction is often misunderstood in practice, because the information need is not overt. But, nevertheless, an information need is present. If a user types python into a web search engine, he might want to know where he can purchase a pet python. Or he might want information on the programming language Python. From a short query, it is very difficult for a system to know what the information need is. But, nevertheless, the user has one, and can judge the returned results on the basis of their relevance to it. To evaluate a system, we require an overt expression of an information need, which can be used for judging returned documents as relevant or non-relevant. At this point, relevance can reasonably be thought of as a scale, with some documents highly relevant and others marginally so.

"Language Modelling approaches the relation between a document and a request by asking: how probable is it that this document generated the request? More strictly the question is: how probable is it that the document, as represented by its index description, generated the request as represented by its indexing description, i.e. the search query? There is no explicit reference to relevance here. However the presumption is that if it is highly probable that the document generated the request, then the document's content is relevant to the information need underlying the user's request" [Jones et al. (2003)].

5.2 Basic IR evaluation model

Researchers share test collections that contain a corpus, queries, and relevance assessments that indicate which documents are relevant to which queries. Because researchers share common resources and guidelines for conducting system evaluations, it is possible to compare search systems and improve search algorithms. Particular evaluation measures indicate how well a search algorithm performs with respect to the number of relevant documents retrieved along with the position of these documents within a ranked list. Common measures include precision, recall, mean average precision, mean reciprocal rank, and discounted cumulative gain. While researchers explore different problems and search strategies, the basic objective of IR system evaluation is to assess search performance, which is usually tied directly to how effectively the system retrieves and ranks relevant information objects.

5.2.1 Precision and Recall

How is system effectiveness measured? The two most frequent and basic measures for information retrieval effectiveness are precision and recall.

Precision is the number of relevant documents a search retrieves divided by the total number of documents retrieved, while the recall is the number of relevant documents retrieved divided by the total number of existing relevant documents that should have been retrieved. These measures were originally intended for set retrieval, but most current research assumes a ranked retrieval model, in which the search returns results in order of their estimated likelihood of relevance to a search query.

5.2.2 MAP and other measures

In recent years, other measures have become more common especially for evaluating ranked results. Most standard in the TREC community is Mean Average Precision (MAP), which provides a single-figure measure of quality across recall levels. Among various evaluation measures, MAP has been shown to have especially good discrimination and stability. For a single information need, Average Precision is the average of the precision value obtained for the set of top k documents existing after each relevant document is retrieved, and this value is then averaged over information needs. That is, if the set of relevant documents for an information need $q_j \in Q$ is $\{d_1, \dots d_{m_j}\}$ and R_{jk} is the set of ranked documents from the top result until you get to document d_k, then

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{jk}) \tag{5.1}$$

When a relevant document is not retrieved at all, the corresponding precision value in the above equation is taken to be 0. For a single information need, average precision approximates the area under the uninterpolated Recall-precision curve, and so MAP is roughly the average area under the Recall-precision curve for a set of queries.

Using MAP, fixed recall levels are not chosen, and there is no interpolation. The MAP value for a test collection is the arithmetic mean of average precision values for individual information needs. (This has the effect of weighting each information need equally in the final reported number, even if many documents are relevant to some queries whereas very few are relevant to other queries.) Calculated MAP scores normally vary widely across information needs when measured within a single system, for instance, between 0.1 and 0.7. Indeed, there is normally more agreement in MAP for an individual information need across systems than for MAP scores for different information needs for the same system. This means that a set of test information needs must be large and diverse enough to be representative of system effectiveness across different queries.

The above measures factor in precision at all recall levels. For many prominent applications, particularly web search, this may not be germane to users. What matters is rather how many good results there are on the first page or the first three pages. This leads to measuring precision at fixed low levels of retrieved results, such as 10 or 30 documents. This is referred to as "Precision at k", for example "Precision at 10". It has the advantage of not requiring any estimate of the size of the set of relevant documents but the disadvantages that it is the least stable of the commonly used evaluation measures and that it does not average well, since the total number of relevant documents for a query has a strong influence on precision at k.

An alternative, which alleviates this problem, is R-Precision. It requires having a set of known relevant documents Rel, from which we calculate the precision of the top Rel documents returned. (The set Rel may be incomplete, such as when Rel is formed by creating relevance judgements for the pooled top k results of particular systems in a set of experiments.) R-precision adjusts for the size of the set of relevant documents: A perfect system could score 1 on this metric for each query, whereas, even a perfect system could only achieve a precision at 20 of 0.4 if there were only 8 documents in the collection relevant to an information need. Averaging this measure across queries thus makes more sense. This measure is harder to explain to naive users than Precision at k but easier to explain than MAP. If there are $|Rel|$ relevant documents for a query, we examine the top $|Rel|$ results of a system, and find that r are relevant, then by definition, not only is the precision (and hence R-Precision) $r/|Rel|$, but the recall of this result set is also $r/|Rel|$. Thus, R-Precision turns out to be identical to the break-even point, another measure which is sometimes used, defined in terms of this equality relationship holding. Like Precision at k, R-Precision describes only one point on the recall-precision curve, rather than attempting to summarize effectiveness across the curve. R-precision turns out to be highly correlated with MAP empirically, despite measuring only a single point on the curve.

5.3 Relevance dimensions in INEX

Since its establishment INEX has defined the relevance of an element according to two graded dimensions, exhaustivity (e) and specificity (s). The former measures how exhaustively an XML element discusses the topic of request, whereas specificity measures how focused the element is on the topic of request [Ogilvie and Lalmas (2006)]. In INEX 2005, exhaustivity is measured using 3+1 levels: highly exhaustive (2), somewhat exhaustive (1), not exhaustive (0) and 'too small' (?). The latter category of 'too small' was introduced to allow assessors to label elements, which although containing relevant information were too small to sensibly reason about their level of exhaustivity. In our experiments we mapped these values to a single binary relevance scale as follows:

$$e = \begin{cases} 1 & \text{if} \qquad e \in 1,2 \\ 0 & \qquad \text{otherwise} \end{cases} \tag{5.2}$$

We ignore elements assessed as 'too small', because assessors may have abused the 'too small' when judging an article, by marking all of the remaining un-assessed elements in an article as 'too small' and continue to the next article.
Specificity was measured on a continuous scale with values in [0,1], where 1 represents a fully specific component (i.e. contains only relevant information). Here, this relevance dimension is only considered in our experiments described in chapter 8.

5.4 Evaluation of XML Retrieval

How to properly evaluate XML retrieval effectiveness is a critical issue in INEX and among the XML retrieval research community. The purpose of an XML retrieval system is to identify and retrieve elements that contain as much relevant information as possible, without also containing a substantial amount of non-relevant information. Over the last years, INEX has been used as an arena to investigate the behaviour of a variety of evaluation metrics. For three years since its beginning in 2002, the inex_eval metric [Gövert and Kazai (2003)] was the official metric used at INEX to evaluate the effectiveness of XML retrieval systems. For a returned element, this metric computes the so-called Precall value, which determines the probability that the retrieved element is relevant [Raghavan et al. (1989)]. There are, however, two weaknesses of this metric: first, the size of the retrieved element is not taken into account during evaluation; and second, the level of overlap - both among the retrieved elements and among the relevant elements found for an INEX topic - is not considered, resulting in possibly inaccurate and misleading evaluation results [Kazai et al. (2004)]. To address some of these problems, the inex_eval_ng metric was proposed as an alternative INEX evaluation metric [Gövert et al. (2003b)]. This metric considers the size and the level of overlap among the retrieved elements, however it too has several shortcomings: first, it is not easy to interpret; second, it assumes that relevant information is uniformly distributed in the element; and last, it treats the two relevance dimensions in isolation by producing separate evaluation scores.

From 2005, the eXtended Cumulated Gain (XCG) metrics were adopted as official INEX metrics [Kazai and Lalmas (2006)], which are extensions of the cumulated gain metrics initially used for document retrieval [Järvelin and Kekäläinen (2002)]. The XCG metrics rely heavily on different combinations of relevance grades from the two INEX 2005 relevance dimensions. These combinations are shown to be difficult to interpret by assessors, which in turn put in question the validity of the reported evaluation results [Trotman (2005)]. Furthermore, when considering the level of overlap among retrieved elements, the XCG metrics use a somewhat ad-hoc methodology in constructing the so-called ideal recall-base [Kazai and Lalmas (2005)]. Here, a dependency normalization function is used to adjust the descendant scores of the ideal elements. The EPRUM metric, which was also used as an alternative evaluation metric at INEX 2005 [Piwowarski (2005)], extends the traditional definitions of precision and recall to model a variety of user behaviours. From INEX 2006, on the INEX relevance definition uses only specificity as a relevance dimension.

5.5 Evaluation metrics used for our experiments

Given the graded relevance (exhaustivity) scale, we measured retrieval quality with the EPRUM (Expected Precision Recall with User Model) metric which was developed within INEX [Piwowarski (2005)] [1] . This metric is based on a more realistic user model which encompasses a large variety of user behaviours. It supposes a set of ideal results. Recall is defined as the ratio of the number of retrieved ideal elements to the number of relevant elements. The ideal run is defined as the run that maximizes the recall for each rank. Precision is defined as the ratio of the length of an ideal run for achieving the

[1] http://inex.is.informatik.uni-duisburg.de/2005/Metrics.html

same level of recall to the size of the retrieved list. The two definitions are generalization of precision and recall in the standard case.

EPRUM is an extension of Recall-precision: It is based on a definition whose special case is the standard Recall-precision as defined in TREC [Voorhees (2003)]. At a given recall level $l(0 < l \leq 1)$, Precision is defined formally as:

$$\text{Precision}(l) = E\left[A_{ind} \cdot \frac{L_l}{S_l}\right] \tag{5.3}$$

Where A_{ind} means the achievement indicator for a recall l and it is used to set the precision to 0 if the recall level cannot be reached for the evaluated list. This is compatible with the classical definition of precision at a given recall where the precision is set to 0 if the list does not contain enough relevant elements. L_l stands for minimum number of consulted list items for achieving a recall l (over all lists) and S_l stands for minimum number of consulted list items for achieving a recall l (system list).

This is just an alternative definition of the precision at a given recall level. In classical IR, if a system retrieves $A+B$ documents, where A is the number of relevant documents and B the number of not relevant documents, then an ideal system would achieve the same recall with a list reduced to A documents. The above definition would result in a precision $\frac{A}{A+B}$ which is the exact definition of precision - the ratio of the number of relevant documents to the number of the retrieved documents. The achievement indicator is used to set the precision to 0 if the recall level can not be achieved; this is also the classical definition of Recall-precision. This definition relates to the expected search length [Cooper (1991)],[Cooper (1995)] and to the Recall-precision as defined in [Raghavan et al. (1989)].

From formula (5.3), the precision at a given recall l (l is the number of ideal units the user wants to see) can be rewritten:

$\text{Precision}(l) = E1 \cdot E2$

Here $E1$ and $E2$ are defined as follows

$$E1 = \sum_i \left(P(F_i^* \geq r) - (P(F_{i-1}^* \geq r)\right) \tag{5.4}$$

$$E2 = \sum_i \frac{1}{i}\left(P(F_i \geq r) - (P(F_{i-1} \geq r)\right) \tag{5.5}$$

where r is the smallest integer superior or equal to l(number of ideal elements) and F_i(resp.F_i^*) is the number of ideal elements found by the user after he consulted the i first ranks of the system list (resp. the ideal list). If the classical case is considered, where an ideal element is retrieved or not at each rank, then $(P(F_i \geq r)$ is either 0 or 1. In this case, it is easy to see that the expected value $E1$(resp.$E2$) is the actual value (or inverse value) of the rank where the r^{th} ideal element has been retrieved.

In order to compute the probability $P(F_i = r)$ needed by formulas 5.3 and 5.4, first the value of the probability $P(x \in \mathcal{S}_i)$ must be estimated, the probability that an ideal element x is seen after the user has considered ranks 1 to i. As the same user model is used in Piwowarski and Gallinari (2004)], a nearly identical formula can be used:

$$P(x \in \mathcal{S}_i) = 1 - \prod_{y} (1 - P(\mathcal{Y} \in \mathcal{C}_i) P(\mathcal{Y} \rightarrow x)) \tag{5.6}$$

where $P(\mathcal{Y} \in \mathcal{C}_i)$ and $P(\mathcal{Y} \rightarrow x)$ are given by the user model instantiation. Given $P(x \in \mathcal{S}_i)$ for any ideal element x and any rank i, it is possible [Piwowarski and Gallinari (2004)] to compute $P(F_i = r)$ and hence $P(F_i \geq r)$:

$$P(F_i = r) = \sum_{\mathcal{A} \subseteq \mathcal{J}, |\mathcal{A}| = r} \prod_{x \in \mathcal{A}} P(x \in \mathcal{S}_i) \prod_{x \in \mathcal{J} \setminus \mathcal{A}} P(x \notin \mathcal{S}_i) \tag{5.7}$$

where $\mathcal{J}$ is the set of ideal elements and the summation is taken over all the subsets $\mathcal{A}$ of cardinality r of the ideal set of element $\mathcal{J}$. The above formula simply enumerates all the cases where exactly r ideal elements are seen by the user. It is possible to compute it in quadratic time with respect to the cardinality of $\mathcal{J}$, or to approximate it using the normal law.

The above formula can be used to compute the precision at any recall level and it is also possible to compute precision at a given rank. EPRUM is implemented in the EVALJ software, along with all the other metrics of INEX. More details about these formulas and their derivations can be found in [Piwowarski (2006)],[Piwowarski and Dupret (2006)].

5.6 Standard test collections

A corpus is a set of documents, or information objects, that searchers access during a study. In traditional IR evaluations, test corpora are fixed and stable. Static corpora facilitate evaluation, as all systems are working with the same collection of information objects. Moreover, they make it possible to create topics that can be searched successfully and provide researchers with some information about the number of topically relevant documents. Many corpora consist of newswire text, others contain hyperlinked text and alternative types of information objects such as webpages, intranet pages, blog postings, or images. Usually test corpora contain only one type of information objects.

Here we describe the two collections we are using for our experiments.

5.6.1 Text Retrieval Conference (TREC)

The U.S. National Institute of Standards and Technology (NIST) has run a large IR test bed evaluation series since 1992. Within this framework, there have been many tracks over a range of different test collections, but the best known test collections are the ones used for the TREC Ad Hoc track during the first 8 TREC evaluations between 1992 and 1999. In total, these test collections comprise 6 CDs containing 1.89 million documents (mainly, but not exclusively, newswire articles) and relevance judgements for 450 information needs, which are called topics and specified in detailed text passages. Individual test collections are defined over different subsets of this data. The early TRECs each consisted of 50 information needs, evaluated over different but overlapping sets of documents. TRECs 6-8 provide 150 information needs over about 528,000 newswire and Foreign Broadcast Information Service articles. Because the test document collections are so large, there are no exhaustive relevance judgements. Rather, relevance judgements of the NIST assessors are available only for the documents that were among the top 100 returned for some system which was entered in the TREC evaluation for which the information need was developed.

In more recent years, NIST has done evaluations on larger document collections, including the 25 million page GOV2 web page collection. From the beginning, the NIST test document collections were orders of magnitude larger than anything available to researchers previously and GOV2 is now the largest Web collection easily available for research purposes. Nevertheless, the size of GOV2 is still more than 2 orders of magnitude smaller than the current size of the document collections indexed by the large web search companies.

5.6.1.1 Documents

The documents are uniformly formatted into an SGML like structure, as can be seen in the following example:

```
<DOC>
<DOCNO>FT911-3</DOCNO>
<PROFILE>AN-BEOA7AAIFT</PROFILE>
<DATE>910514
</DATE>
<HEADLINE>
```

```
FT 14 MAY 91 / International Company News:  Contigas plans
DM900m east German project
</HEADLINE>
<BYLINE>
By DAVID GOODHART
</BYLINE>
<DATELINE>
BONN
</DATELINE>
<TEXT>
CONTIGAS, the German gas group 81 per cent owned by the
utility Bayernwerk, said  yesterday  that it  intends   to invest
DM900m (Dollars 522m) in the next four years to build a new
gas distribution system in the east German state of Thuringia.
</TEXT>
</DOC>
```

All documents have beginning and end markers, and a unique DOCNO id field. Additionally other fields taken from the initial data appeared, but these varied widely across the different sources. The documents also have different amounts of errors, which were not checked or corrected. Not only would this have been an impossible task, but the errors in the data provided a better simulation of the real-world task.

5.6.1.2 Topics

The topics were designed to mimic a real user's need, and were written by people who are actual users of a retrieval system. TREC distinguishes between a statement of information need (the topic) and the data structure that is actually given to a retrieval system (the query) [Voorhees and Buckland (2005)],[Voorhees (2000)]. The TREC test collections provide topics to allow a wide range of query construction methods to be tested and also to include a clear statement of what criteria make a document relevant. The format of a topic statement has evolved since the earliest TRECs, but it has been stable since TREC-5 (1996). A topic statement generally consists of four sections: an identifier, a title, a description, and a narrative. The different parts of the TREC topics allow researchers to investigate the effect of different query lengths on retrieval performance. TREC distinguishes among two major categories of query construction techniques, automatic methods and manual methods. An automatic method is a means of deriving a query from the topic statement with no manual intervention whatsoever, a manual method is anything else. The definition of manual query construction methods is very broad, ranging from simple tweaks to an automatically derived query, though manual construction of an initial query, to multiple query reformulations based on the document sets retrieved. Since these methods require radically different amounts of (human) effort, care must be taken when comparing manual results to ensure that the runs are truly comparable. TREC topic statements are created by the same person who performs the relevance assessments for that topic (the assessor). Usually, each assessor comes to NIST with ideas for topics based on his or her own interests, and searches the document collection using NIST's PRISE system to estimate the likely number of relevant documents per candidate topic. The NIST TREC team selects the final set of topics from among these candidate topics based on the estimated number of relevant documents and balancing the load across assessors.

Topic example:

```
<top>
<num> Number :503
<title> Vikings in Scotland

<desc> Description:
     What hard evidence proves that the Vikings visited or lived in
     Scotland?

<narr> Narrative:
     A document that merely states that the Vikings visited or lived in
     Scotland is not relevant.  A relevant document must mention the
     source of the information, such as relics, sagas, runes or other
     records from those times.
   </top>
```

5.6.1.3 Qrels (query-relevance set)

A qrels file stores relevant results for queries, i.e. set of judgments for each topic. According to the TREC site [1].

"The format of a qrels file is as follows:

TOPIC ITERATION DOCUMENT RELEVANCY

where TOPIC is the topic number,
ITERATION is the feedback iteration (almost always zero and
not used),

DOCUMENT is the official document number that corresponds
to the "docno" field in the documents, and

RELEVANCY is a binary code of 0 for not relevant and 1 for relevant.

[1] http://trec.nist.gov/data/qrels_eng/index.html

Sample Qrels File:

```
1 0 AP880212-0161 0
1 0 AP880216-0139 1
1 0 AP880216-0169 0
1 0 AP880217-0026 0
1 0 AP880217-0030 0
```

The order of documents in a qrels file is not indicative of relevance or degree of relevance. Only a binary indication of relevant (1) or non-relevant (0) is given. Documents not occurring in the qrels file were not judged by the human assessor and are assumed to be irrelevant in the evaluations used in TREC. The human assessors are told to judge a document relevant if any piece of the document is relevant (regardless of how small the piece is in relation to the rest of the document)."

5.6.1.4 Test collection used for our TREC experiments

The TREC part of our experiments was conducted on the AP subset of the TREC collection which consists of 240,000 AP documents.

Queries are based on TREC topics 51-100 and 101-150 [Harman (1995)], respectively and as test queries, we used the description fields. For both documents and queries, terms are stemmed (using the Porter stemmer [Porter (1980))], and stop words (the TREC "common words") are removed. The relevance judgments are the standard TREC relevance judgements [Harman (1995)], documents which have no judgement are treated as irrelevant and we used the trec-eval package for the evaluation.

5.6.2 INEX

The XML document collection used by the INEX until 2006 was comprised of IEEE Computer Society research publications, converted in XML format. However, two different collection sizes were used during these four years. More specifically, a collection comprising 12,107 IEEE Computer Society articles, published in the period between 1997-2002 with approximately 500MB of data, was used in 2002, 2003, and 2004; an expanded collection comprising 16,819 IEEE Computer Society articles, published in the period between 1997-2004 with approximately 735MB of data, was used in 2005.

5.6.2.1 Documents

An XML document in INEX collection consists of front-matter (containing title, names of authors, and abstract), body (containing the document text enclosed in sections, subsections, or paragraphs), and back-matter (containing bibliography and author). Appendix A contains an example which shows the structure of one of the documents from the INEX 2005 collection.

5.6.2.2 Topics

Topics are made of several parts, these parts explain the same information need, but for different purposes.

The following shows examples for INEX 2005 topics.

$< inex\_topictopic\_id = ”202” query\_type = ”CO + S” ct\_no = ”1” >$

$< InitialTopicStatement >$

I'm interested in knowing how ontologies are used to encode knowledge in real world scenarios. I'm writing a report on the use of ontologies. I'm particularly interested in knowing what sort or concepts and relations people use in their ontologies. $< /InitialTopicStatement >$

$< title >$

ontologies case study$< /title >$

$< castitle >$

//article[about(., ontologies)]//sec[about(., ontologies case study)]$< /castitle >$

$< description >$

Case studies in the use of ontologies$< /description >$

$< narrative >$

I'm writing a report on the use of ontologies. I'm interested in knowing how ontologies are used to encode knowledge in real world scenarios. I'm particularly interested in knowing what sort or concepts and relations people use in their ontologies. I'm not interested in general ontology frameworks or technical details about tools for ontology creation or management. An example relevant result contains a description of the real world phenomena described by the ontology and also lists some of the concepts used and relations between concepts. $< /narrative >$

$< /inex\_topic >$

$< inex\_topictopic\_id = ”203” query\_type = ”CO + S” ct\_no = ”5” >$

$< InitialTopicStatement >$

Code signing is an approach for authenticating code based on public-key cryptography and digital signatures. The digital signature lets a user of the application code determine which particular key the code was signed with, and further ensures that the code has not been tampered with since it was signed. I am working in a company that authenticates a wide range of web database applications from different software vendors. I am looking for documents or document components that describe the code signing and verification approach.$< /InitialTopicStatement >$

$< title >$

code signing verification$< /title >$

$< castitle >$

//sec[about(., code signing verification)]$< /castitle >$

$< description >$

Find documents or document components, most probably sections, that describe the

approach of code signing and verification.$<$ */description* $>$
$<$ *narrative* $>$
I am working in a company that authenticates a wide range of web database applications from different software vendors. My work mainly focuses on the following two activities: checking whether the code that originates from a software vendor is authentic and properly signed, and checking whether the code has been tampered with since it was signed. I am looking for documents or document components that describe the approach of code signing and verification. To be relevant, a document or document component must describe the whole process of code signing and verification, which means ensuring that programs and program components have been created by trusted entities (by validating both the digital signature and the corresponding certificate), and that the programs have been received without tampering (by checking the main integrity of the program). Description of implementations of various approaches to code signing (such as Microsoft's Autheticode and Sun's JAR signing) are also relevant. A document or document component that only describe CRC-type integrity check of received programs will be considered only marginally relevant. Relevant information about this topic can probably be found within the section components of the documents in the collection.$<$ */narrative* $>$
$<$ */inex_topic* $>$

5.6.2.3 Test collection used for our INEX experiments

We used the INEX collection [Fuhr et al. (2006a)], version 1.9. This collection consists originally of 16,819 journal articles in XML format, comprising 764 MB of data. For our experiments, we regarded each XML element as an independent document, thus leading to a collection of 21.6 million documents with a collection size of more than 253 million words. For our experiments, we considered the 29 queries from INEX 2005 (version 003) and as a search requests, our queries were created using terms only in the <title> parts of the topic (the so-called CO queries), which are free text queries. For both documents and queries, terms are stemmed (using the Porter stemmer [Porter (1980))], and stop-words removed using the stop-word list that comes with the English version on the Snowball stemer.[1]. The relevance judgments are the the official adhoc 2005-assessments-v7.0.

[1] http://snowball.tartarus.org/algorithms/english/stop.txt

6 Experiments with Divergence From Randomness (DFR)

Divergence From Randomness (DFR) models are among the best performing IR models. DFR is a kind of general language model, and it had been applied to XML IR successfully in our group [Abolhassani and Fuhr (2004)]. We also worked on an extension of this model. Here we present it in a simplified way along with experimental results.

6.1 Divergence From Randomness (DFR)

The Divergence from Randomness (DFR) [Amati and van Rijsbergen (2002b)] paradigm is a generalization of one of the very first models of Information Retrieval, Harter's 2-Poisson indexing-model. The 2-Poisson model (2.1.3) is based on the hypothesis that the level of treatment of the informative words is witnessed by an elite set of documents, in which these words occur to a relatively greater extent than in the rest of the documents.

On the other hand, there are words, which do not possess elite documents, and thus their frequency follows a random distribution, that is the single Poisson model. Harter's model was first explored as a retrieval model by Robertson, Rijsbergen and Porter [Robertson et al. (1981)]. Successively it was combined with the standard probabilistic model by Robertson and Walker [Robertson and Walker (1994)] and gave birth to the family of the BMs IR models (among them there is the well known BM25 which is at the basis of the Okapi system [Robertson et al. (1992)]).

The framework is used for deriving probabilistic models of IR. These models are non-parametric models of IR as obtained in the language model approach. The term weighting models are derived by measuring the divergence of the actual term distribution from that obtained under a random process. There are two basic assumptions underlying this approach:

1. Words which bring little information are randomly distributed on the whole set of documents. One can provide different basic probabilistic models, with probability distribution $prob1$, that define the notion of randomness in the context of IR.
2. If one restricts statistics to the set of all documents in which a term occurs, the "elite" set, then one can derive a new probability $prob2$ of the occurrence of the word within a document with respect to its elite set.

Based on these ideas, the weighting formula for a term in a document is the product of the following two factors:

1. $prob1$ is used for measuring the information content of the term in a document, and $(-\log_2 prob1)$ gives the corresponding amount of information.
2. $prob2$ is used for measuring the information gain of the term with respect to its "elite" set (the set of all documents in which the term occurs). The less the term is expected in a document with respect to its frequency in the elite set, measured by the counter-probability $(1 - prob2)$, the more information is gained with this term.

Now the weight of a term in a document is defined as

$w_{DFR}(t) = (\mathit{1} - \mathit{prob2}) \cdot (-\log_{\mathit{2}} \mathit{prob1}) = \mathit{Inf2} \cdot \mathit{Inf1}$

Using various approximations, this finally leads to the following formulas:
The approximation of the binomial with the divergence

$$Inf1a = tf \cdot \log_2(\frac{tf}{\lambda}) + (\lambda + \frac{1}{12tf} - tf) \cdot \log_2 e + 0.5 \log_2(2\pi \cdot tf) \tag{6.1}$$

The Geometric as limiting form of the Bose-Einstein model:

$$Inf1b = -\log_2 \frac{1}{1+\lambda} - tf \cdot \log_2 \frac{\lambda}{1+\lambda} \tag{6.2}$$

Based on Laplace's law of succession:

$$Inf2a = \frac{1}{tf + 1} \tag{6.3}$$

Regarding the ratio of two Bernoulli processes yields

$$Inf2b = \frac{F+1}{n \cdot (tf + 1)} \tag{6.4}$$

Here we use the following notations:

N number of documents in the collection,

tf term frequency within the document,

n size of the elite set of the term,

F term frequency in elite set,

$\lambda = F/N$.

6.2 Experiments with the TREC collection

For comparing the DFR approach with other models, we considered the following IR systems:

Lemur: Lemur [1], is an information retrieval toolkit designed with language modeling in mind. Lemur is written in C++ and C for use under UNIX and Windows NT. It supports two types of indexes: one storing a bag-of words representations for documents, the other storing term location information and supports several retrieval algorithms. The primary retrieval model is a unigram language-modeling algorithm based on Kullback-Leibler divergence [Cover and Thomas (1991)]. Also included is the OKAPI retrieval algorithm [Robertson et al. (1998)] and a dot-product function using $tf \cdot idf$ weighting.[2] Our experiments were based on the Kullback-Leibler (KL) divergence algorithm where documents are ranked according to the negative of the divergence of the query's language model from the document's language model (see section 3.5).

HyREX: HyREX [3], is the Hyper-media Retrieval Engine for XML [Abolhassani et al. (2002)]. It offers explicit and implicit links to the user. Explicit links are specified within the documents, usually by means of XML linking standards, such as Xlink and XPointer. Implicit links are intrinsic to information structures which HyREX derives from XML document collections. HyREX offers search facilities for text, but also for other media than text, at least conceptually. It allows users to explore all kinds of information structure available through XML; besides retrieval in XML documents it allows for browsing and searching the domains of attributes of XML documents as well as schema information given for example by the DTD of a documents. HyREX allows retrieval under consideration of content and structure inherent in XML documents. The physical layer HyPath deals with efficient access paths for retrieval, while the logical layer deals with the XIRQL [Fuhr and Großjohann (2002)] query language. On top of these layers is HyGate, the user interface to HyREX applications. The actual retrieval and weighting function is specified in [Gövert et al. (2003a)].

[1] http://www.lemurproject.org

[2] http://www.cs.cmu.edu/ lemur/1.0/tfidf.ps

[3] http://www.is.informatik.uni-duisburg.de/projects/HyREX/index.html

PIRE: PIRE [Nottelmann (2005)], is a probabilistic IR engine. For both document indexing and retrieval, PIRE makes heavy use of probabilistic Datalog, a probabilistic extension of predicate Horn logics. Using such a logical framework together with probability theory allows for defining and using data types(e.g. text, names, numbers), different weighting schemes (e.g. normalised tf, $tf \cdot idf$ or BM25) and retrieval functions (e.g. uncertain inference, language models).

Our DFR experiments used PIRE, and the results were compared to Kullback-Leibler (KL) implemented in Lemur, HyREX and BM25 implemented in PIRE. In the experiments below, we tested the DFR weighting model on the TREC collection, with the following weighting formulas:

DFR a/a :
$weight(d,t) = inf2a(d,t) \cdot inf1a(d,t)$

DFR a/b :
$weight(d,t) = inf2a(d,t) \cdot inf1b(d,t)$

DFR b/a :
$weight(d,t) = inf2b(d,t) \cdot inf1a(d,t)$

DFR b/b :
$weight(d,t) = inf2b(d,t) \cdot inf1b(d,t)$

The results of our experiments summarized in (table 6.1 and figure 6.1), show that DFR a/b is the best among the DFR formulas. Although we used the DFR models in a simplified way only, we found that they are very effective. They have the advantage of being nonparametric models, which means that they do not need to be supported by any form of data-driven methodology, such as learning of parameters from a training collection, or using data smoothing techniques. The other important feature is that different choices of probability distributions can be used, such as the binomial distribution or Bose-Einstein statistics. All the models show comparable results to BM25 which is frequently used by many participants of TREC.

Table 6.1: MAP, P@5, P@10, P@20 - TREC collection

Method	MAP	Prec at 5	Prec at 10	Prec at 20
KL	0.0921	0.114	0.090	0.077
HyREX	0.1809	0.220	0.182	0.140
BM25	0.2341	0.278	0.227	0.168
DFR a/a	0.2220	0.262	0.208	0.158
DFR a/b	0.2352	0.272	0.224	0.167
DFR b/a	0.2097	0.238	0.189	0.158
DFR b/b	0.2333	0.262	0.221	0.167

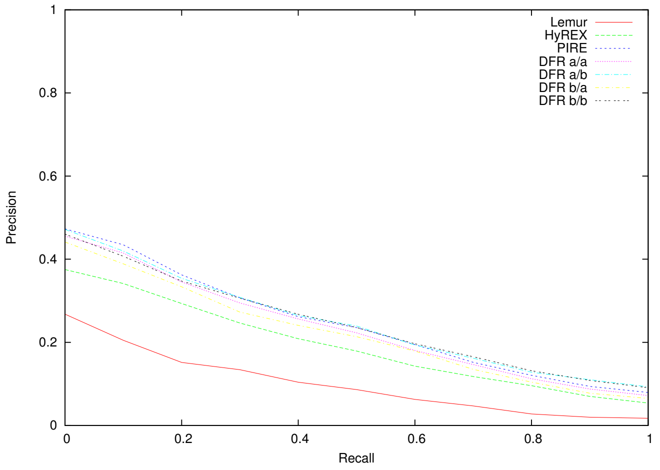

Figure 6.1: Recall-precession curves using TREC collection for different models

6.3 Experiments with the INEX XML collection

Abolhassani and Fuhr [Abolhassani and Fuhr (2004)] applied the DFR directly for XML retrieval, but the retrieval quality was lower than the best results known before.
Next, they extended the DFR model by a normalisation component which takes into account the hierarchical structure of XML documents and this approach has improved the results.

$$Inf2_{Abo} = \frac{1}{\frac{h(d)}{\alpha} \cdot tfn + 1} \tag{6.5}$$

Here α is a parameter to be chosen and $h(d)$ is the height(or level) of an index node relative to the root node (which has $h = 1$).

Following [Abolhassani and Fuhr (2004)] for developing a new model for XML retrieval, and to test the possible parameters for out degree and the level (or the height, which was used by the last equation of Abolhassani & Fuhr) we developed several new models.

In order to see if the outdegree of an XML node has any effect, we assume the probability of observing term in specific element at level h is $\frac{1}{\nu^h}$. So we have

$$prob2' = \frac{1}{\nu^h} = 1 - inf2 \tag{6.6}$$

$$inf2' = 1 - \frac{1}{\nu^h} \tag{6.7}$$

Using Amati's definition

$$inf2 = \frac{1}{tf + 1} = 1 - prob2, \tag{6.8}$$

we define:

$$inf2a = inf2 + inf2' = \frac{1}{tf + 1} + (1 - \frac{1}{\nu^h}) \tag{6.9}$$

$$inf2b = 1 - prob2 \cdot prob2' = 1 - (\frac{tf}{tf + 1} \cdot \frac{1}{\nu^h}) \tag{6.10}$$

With $inf1a$ and $inf1b$ as defined by Amati,
we now consider the combinations as

$waa = inf2a \cdot inf1a$
$wab = inf2a \cdot inf1b$
$wba = inf2b \cdot inf1a$
$wbb = inf2b \cdot inf1b$

which results in the following formulas:

$$\begin{aligned}
waa &= \left[\frac{1}{tfn + 1} + (1 - \frac{1}{\nu^h})\right] \cdot \left[tfn \cdot \log_2(\frac{tfn}{\lambda}) + (\lambda + \frac{1}{12tfn}) \cdot \log_2 e + 0.5 \log_2(2\pi \cdot tfn)\right] && (6.11)\\
wab &= \left[\frac{1}{tfn + 1} + (1 - \frac{1}{\nu^h})\right] \cdot \left[- \log_2 \frac{1}{1 + \lambda} - tfn \cdot \log_2 \frac{\lambda}{1 + \lambda}\right] && (6.12)\\
wba &= \left[1 - ((\frac{tf}{tf + 1}) \cdot \frac{1}{\nu^h})\right] \cdot \left[tfn \cdot \log_2(\frac{tfn}{\lambda}) + (\lambda + \frac{1}{12tfn}) \cdot \log_2 e + 0.5 \log_2(2\pi \cdot tfn)\right] && (6.13)\\
wbb &= \left[1 - ((\frac{tf}{tf + 1}) \cdot \frac{1}{\nu^h})\right] \cdot \left[- \log_2 \frac{1}{1 + \lambda} - tfn \cdot \log_2 \frac{\lambda}{1 + \lambda}\right] && (6.14)
\end{aligned}$$

Here we use the following notations:

N number of documents in the collection

tf term frequency within the document

F term frequency in elite set

$\lambda = F/N$

ν = outdegree

h height (level)

tfn normalized term frequency

Depending on the parameter β, we have two different formulas for tfn

$\beta = -1$:

$$tfn = tf \cdot \log_2 \left(1 + \frac{avl}{dl}\right) \tag{6.15}$$

$\beta > -1$:

$$tfn = \frac{tf}{dl(\beta + 1)} \cdot (\exp((\beta + 1)\log_2(dl + avl)) - \exp((\beta + 1)\log_2(dl))) \tag{6.16}$$

Here we use the additional notations:

dl document length

avl the average length of documents

In the following, we only use the first definition (for β= -1), since it led to better results in our experiments.

We tested the different weight equations to see which of them is best for ranking and what is the effect of the outdegree. Table 6.2 and figure 6.2 show our results. As can be seen, the *wab* model outperformed the other formulas . Comparing to our last experiments (table 6.1 and figure 6.1), here results are worse than that of the previous models. However, [Abolhassani and Fuhr (2004)] still achieved better retrieval performance, so further enhancement of the models may lead to better results. Due to these disappointing results, we decided to give up on the DFR approach, and consider standard language models instead.

Table 6.2: Results for new DFR approach (INEX) collection

Model	MAP	Prec at 5	Prec at 10	Prec at 20
wab	0.0641	0.346	0.274	0.241
wbb	0.0605	0.193	0.183	0.178
waa	0.0300	0.169	0.159	0.172
wba	0.0252	0.145	0.141	0.147

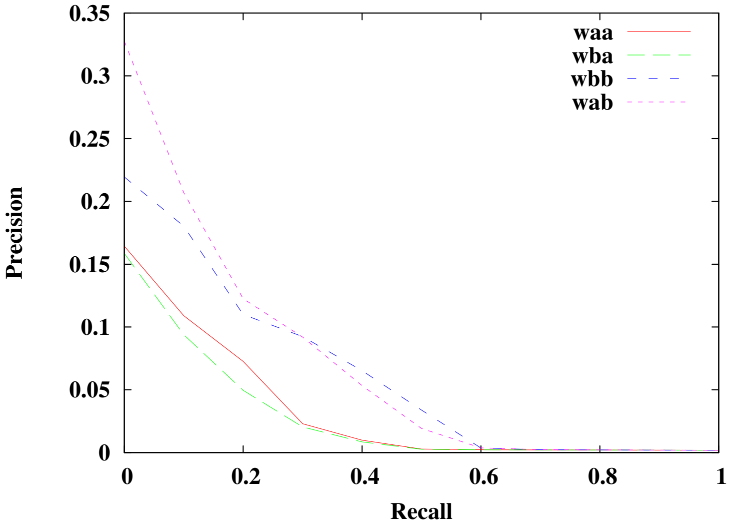

Figure 6.2: Recall-precision curves for new DFR approach (INEX) collection

7
Language Models and Smoothing Methods for Collections with Large Variation in Document Length

Here we present a new language model based on an odds-like formula, which explicitly incorporates document length as a parameter. Furthermore, a new smoothing method called exponential smoothing is introduced, which can be combined with most language models. We present experimental results for various language models and smoothing methods on a collection with large document length variation, and show that our new methods compare favorably with the best approaches known so far. [Abdulmutalib and Fuhr (2008)]

7.1 Introduction

Since the first language models for information retrieval were presented [Ponte and Croft (1998b)],[Hiemstra (1998)],[Miller et al. (1999)], a large variety of models of this kind have been proposed. However, with the exception of [Losada and Azzopardi (2008)], little attention has been paid to the influence of document length, and only a few approaches have considered this parameter explicitly.

In the next section, we present a new language model which includes document length as a genuine parameter. We start along the lines of the classic Zhai/Lafferty [Zhai and Lafferty (2001b)] model and present a probability model and an odds model as variations of this basic model. Section 7.3 introduces a new smoothing method for combining the relative term frequencies in the current document and the whole collection into a single probability estimate. As an alternative to this smoothing method, we also consider the classic smoothing methods regarded by Zhai and Lafferty, and then present experimental results for the INEX collection in Section 7.5, where we regard each XML element as a document, thus having a collection with a large variation of document lengths.

7.2 An Odds model

[Fuhr (2001b)] shows that language models can be interpreted in terms of uncertain inference, and that Hiemstra's model [Hiemstra (1998)] regards the probability $P(q \rightarrow d)$ $= P(d|q)$ of the query implying the document. In contrast, the basic probability model from chapter 3 focuses on the implication in the reverse direction, i.e. $P(d \rightarrow q) = P(q|d)$. In order to develop a language model formula with explicit consideration of document length, we start from Hiemstra's approach, but use an odds-like formula comparing the actual document under consideration to the 'average' document from the collection. Thus, we regard the probability that the query implies the document, which we divide by the probability $P(q \rightarrow \bar{d}) = P(\bar{d}|q)$ that the query implies an arbitrary document $\bar{d}$ different from d. Applying Bayes' theorem and the standard independence assumptions, we get:

$$\begin{aligned}
\frac{P(d|q)}{P(\bar{d}|q)} &= \frac{P(q|d)}{P(q|\bar{d})} \cdot \frac{P(d)}{P(\bar{d})} && (7.1)\\
&= \prod_{t_i \in q^T} \frac{P(t_i|d)}{P(t_i|\bar{d})} \frac{P(d)}{P(\bar{d})} && (7.2)\\
&= \prod_{t_i \in q^T \cap d^T} \frac{P_s(t_i|d)}{P_s(t_i|\bar{d})} \prod_{t_i \in q^T - d^T} \frac{P_u(t_i|d)}{P_u(t_i|\bar{d})} \cdot \frac{P(d)}{P(\bar{d})} && (7.3)
\end{aligned}$$

In addition to the parameters defined for the probability model, we have the following probabilities here:

$P(\bar{d})$ Probability that an arbitrary document $\neq d$ implies an arbitrary query,

$P_s(t_i|\bar{d})$ Probability that an arbitrary document $\neq d$ implies term t_i, given that t_i occurs in that document,

$P_u(t_i|\bar{d})$ Probability that an arbitrary document $\neq d$ implies term t_i, given that t_i does not occur in that document,

7.3 Smoothing methods

As an alternative to the smoothing methods as described before, we developed a new method which would also allow us to combine it with logistic regression.

7.3.1 Exponential smoothing

As an estimate of $P_s(t_i, d)$ we propose an exponential formula for combining $P_{ml}(t_i|d)$ and $P_{avg}(t_i|C)$. In a similar way, we estimate $P_u(t_i|d)$ as a function of $P_{avg}(t_i|C)$. More precisely, our estimates are (where the additional subscript 'e' refers to the smoothing method):

$$\begin{aligned} P_{s,e}(t_i|d) &= P_{ml}(t_i|d)^{\alpha} \cdot P_{avg}(t_i|C)^{1-\alpha} && (7.4) \\ P_{u,e}(t_i|d) &= P_{avg}(t_i|C)^{\beta} && (7.5) \end{aligned}$$

Here α and β are smoothing factors.

In the same way, we estimate $P_{s,e}(t_i|\bar{d})$ and $P_{u,e}(t_i|\bar{d})$ with μ and δ smoothing factors.

$$\begin{aligned} P_{s,e}(t_i|\bar{d}) &= P_{ml}(t_i|d)^{\mu} \cdot P_{avg}(t_i|C)^{1-\mu} && (7.6) \\ P_{u,e}(t_i|\bar{d}) &= P_{avg}(t_i|C)^{\delta} && (7.7) \end{aligned}$$

With these estimation formulas, the term-specific factors in equation (7.3) can be rewritten as:

$$\frac{P_{u,e}(t_i|d)}{P_{u,e}(t_i|\bar{d})} = P_{avg}(t_i|C)^{\beta-\delta} = P_{avg}(t_i|C)^{\gamma} \tag{7.8}$$

(where $\gamma = \beta - \delta$), and

$$\begin{aligned} \frac{P_{s,e}(t_i|d)}{P_{s,e}(t_i|\bar{d})} &= P_{ml}(t_i|d)^{\alpha-\mu} \cdot P_{avg}(t_i|C)^{-\alpha+\mu} && (7.9) \\ &= P_{ml}(t_i|d)^{\omega} \cdot P_{avg}(t_i|C)^{-\omega} && (7.10) \end{aligned}$$

(where $\omega = \alpha - \mu$).

Applying exponential smoothing to our odds model, we get the retrieval function

$$\begin{aligned} \rho_{o,e}(q,d) &= \prod_{t_i \in q^T \cap d^T} P_{ml}(t_i|d)^{\omega} \cdot P_{avg}(t_i|C)^{-\omega} \\ &\quad \cdot \prod_{t_i \in q^T - d^T} P_{avg}(t_i|C)^{\gamma} \cdot \frac{P(d)}{P(\bar{d})} && (7.11) \end{aligned}$$

$$= \prod_{t_i \in q^T \cap d^T} \left(\frac{P_{ml}(t_i|d)}{P_{avg}(t_i|C)} \right)^{\omega} \cdot \prod_{t_i \in q^T - d^T} P_{avg}(t_i|C)^{\gamma} \cdot \frac{P(d)}{P(\bar{d})} \tag{7.12}$$

We finally have only two smoothing factors (ω and γ) and we have the additional parameters $P(d)$ and $P(\bar{d})$. The former denotes the probability that document d generates a random query, while the latter denotes the same probability for an arbitrary document different from d. In a similar way, the probability model with exponential smoothing yields

$$\rho_{p,e}(q,d) = \prod_{t_i \in q^T \cap d^T} \frac{P_{ml}(t_i|d)^{\alpha}}{P_{avg}(t_i|C)^{\beta+\alpha-1}} \prod_{t_i \in q^T} P_{avg}(t_i|C)^{\beta} \tag{7.13}$$

Since the second factor is independent of the specific document, we can also ignore it when we are only interested in the ranking of the documents.

Losada and Azzopardi [Losada and Azzopardi (2008)] studied different Language Modelling smoothing strategies from a document length retrieval perspective and showed that the document length retrieval pattern is of major importance in language modelling for information retrieval. In some initial experiments, we also noticed that document length plays an important role and significantly improves the retrieval quality. For this reason, we decided to regard a variant of the probability model which incorporates document length, thus leading to the retrieval function

$$\rho^{d}_{p,e}(q,d) = \prod_{t_i \in q^T \cap d^T} \frac{P_{ml}(t_i|d)^{\alpha}}{P_{avg}(t_i|C)^{\beta+\alpha-1}} \cdot \frac{P(d)}{P(\bar{d})} \tag{7.14}$$

As a first approximation, we assume that the probability $P(d)$ is proportional to document length, which we use as estimates in (7.11) and (7.14) for the experiments described below.

7.4 Comparison with the Ponte and Croft model

Ponte and Croft assume that the user of an IR system "has a prototypical document in mind" [Ponte and Croft (1998a)] and knows good terms that are likely to occur in such an ideal document to a varying degree of accuracy. The model uses as a ranking function the estimate of the probability that the user derives the given query when each particular document is postulated as an ideal document. The user is supposed to know what terms are more likely than others to be found in an ideal document which he seeks. While we use the maximum likelihood estimate for the document language model, Ponte and Croft estimate the language model by using a geometric risk function.

Their model estimates the probability of q being generated when d is assumed to be an ideal document as follows:

$$P(q|d) = \prod_{t \in q^T} P(t|d) \cdot \prod_{t \notin q^T} (1.0 - P(t|d)) \tag{7.15}$$

The estimation technique used in the their model is shown below:

$$P(t|d) = \begin{cases} P_{ml}(t|d)^{(1.0-R_{t,d})} \cdot P_{avg}(t|t \in d')^{R_{t,d}} & \text{if} \quad tf(t,d) > 0 \\ \frac{cf(t)}{cs} & \text{otherwise} \end{cases} \tag{7.16}$$

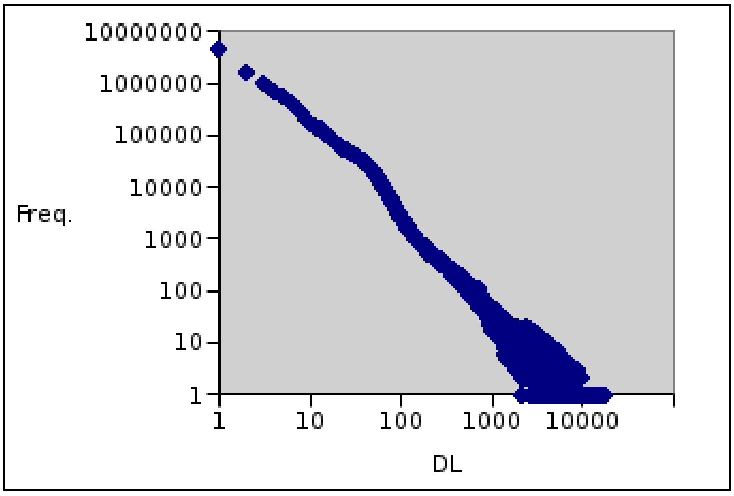

Figure 7.1: Distribution of document lengths in INEX

Where

$$R_{t,d} = \left(\frac{1.0}{(1.0 + \bar{tf}(t,d))}\right) \cdot \left(\frac{\bar{tf}(t,d)}{(1.0 + \bar{tf}(t,d))}\right)^{tf(t,d)} \tag{7.17}$$

Here $cf(t)$ is the raw count of term t in the collection and cs is the raw collection size or the total number of tokens in the collection, $\bar{tf}$ is the mean term frequency of term t in documents.

The Ponte and Croft model combines two estimates through a risk function $R_{t,d}$. The value of this function becomes larger as $tf(t;d)$ (term frequency) moves away from its expected value, $\bar{tf}(t;d)$. Therefore, depending on the risk of using a maximum-likelihood estimate, the model reverts back to the more reliable estimate of $P(t|d)$. Probability estimates computed from the entire collection statistics are used for terms which do not occur in each document.

Furthermore, Ponte and Croft assumed $P(d)$ to be uniform, so it does not affect document ranking, which is the simplest case. In our model, $P(d)$ is proportional to the length of a document which we have found to be of major importance.

7.5 Experimental results

First, we regarded the effect of considering document length. Figure 7.1 shows the distribution of document lengths in our test collection. Here document length ranges from 1 to 17784. We obviously have a linear relationship between the logarithms of document lengths and frequency. This is certainly a kind of document length distribution which can only be found in the special setting we are regarding here, namely retrieval of XML elements. On the other hand, this situation also serves as a good test case for investigating the influence of document length variation on the retrieval quality of language models. For the odds and the probability model, Tables 7.1 and 7.2 show the best results with and without considering document length (using exponential smoothing); for the odds model, the factor $\frac{P(d)}{P(\bar{d})}$ was omitted from the retrieval formula for $\rho_{o,e}$ when document length was ignored; in the case of the probability model, the functions for $\rho^d_{p,e}$ and $\rho_{p,e}$ were compared. The experimental results show huge performance differences for both kinds of models. So document length is an important factor for achieving good retrieval results when dealing with collections of varying document size.

In a second series of experiments, we investigated the effect of exponential smoothing on the performance of the odds and the probability model. For this purpose, we varied the values of the smoothing parameters between 0 and 1 and performed a large number of runs. The MAP values of these experiments are shown in Tables 7.3, 7.4 and figures 7.2, 7.3. Our results indicate that the retrieval performance is sensitive to the values of the smoothing parameters.

For the probability model, the best results were achieved when β approaches 1 and α takes values between 0.4 and 0.8. For the odds model, the retrieval performance was the highest for $\gamma = 0.2$ and ω between 0.4 and 0.6.

Finally, we compared the best results of our new models and smoothing method with those of the Zhai/Lafferty model in combination with different smoothing methods.

The results depicted in table 7.5, figure 7.5, figure 7.6 indicate that the probability and the odds model yield their best results when combined with the exponential smoothing, and they even outperform the Zhai/Lafferty model. For the latter, the best results were achieved in combination with Bayesian Dirichlet smoothing. We think that this outcome is due to the fact that Bayesian Dirichlet is the only smoothing method which explicitly considers document length. In contrast, other smoothing methods lead to very poor performance figures for the Zhai/Lafferty model. So this model should only be used in combination with Bayesian Dirichlet smoothing when being applied to collections with varying document size.

The results of the three best combinations (probability and odds model with exponential smoothing, Zhai/Lafferty with Bayesian Dirichlet) are also illustrated in the precision-rank curve shown in Figure 7.4. The results indicate much better performance for our models than for the Zhai/Lafferty model and we believe that this is a promising results for our models. With variants of the document length parameter and (possibly) document-specific smoothing, there are still possibilities for further improvement.

Table 7.1: Best results for odds model ($\gamma = 0.2$) with and without using document length

Omega	Normal model using DL	Ignoring DL
0	0.006	0.016
0.1	0.014	0.002
0.2	0.038	0.002
0.3	0.064	0.002
0.4	0.078	0.002
0.5	0.080	0.002
0.6	0.076	0.002
0.8	0.068	0.002
0.9	0.063	0.002

Table 7.2: Best results for prob. model ($\beta = 1$) with and without using document length

Alpha	Normal model using DL	Ignoring DL
0	0.006	0.018
0.1	0.020	0.027
0.3	0.059	0.027
0.4	0.076	0.027
0.5	0.079	0.027
0.6	0.076	0.027
0.8	0.070	0.027
0.9	0.063	0.027

7.6 Cross validation

In the experiments described above, the parameters α, γ and ω were tuned on the same collection where we evaluated the retrieval quality. For getting more valid results, we performed additional experiments applying cross-validation in the following way: We used the leave-one-out method at the query level, by tuning the parameters for $n - 1$ queries and tested on the remaining query. This process was repeated n times, and then we computed the retrieval quality for the n queries. For the probability model we found that MAP is 0.108 and all the queries gave their best results when α=0.5. For the odds model the MAP was 0.223 and nearly all the queries yield the best results for ω=0.5 except, for topic 213 where the best results were reached for ω=0.6.

Table 7.3: Influence of α and β parameters on MAP when using prob. model

α	β				
	0.1	0.2	0.5	0.9	1
0.1	0.003	0.003	0.003	0.004	0.020
0.3	0.003	0.003	0.003	0.013	0.059
0.4	0.003	0.003	0.003	0.037	**0.076**
0.5	0.003	0.003	0.003	0.057	**0.079**
0.6	0.003	0.003	0.003	0.063	**0.076**
0.8	0.003	0.003	0.007	0.061	0.070
0.9	0.003	0.003	0.014	0.060	0.063

Table 7.4: Influence of ω and γ parameters on MAP when using odds model

Omega	Gamma						
	0	0.1	0.2	0.3	0.5	0.8	0.9
0	0.005	0.003	0.006	0.006	0.008	0.012	0.013
0.1	0.011	0.013	0.014				
0.2	0.033	0.036	0.038				
0.3	0.060	0.062	0.064				
0.4	0.076	0.077	**0.078**		0.066		0.052
0.5	0.079	0.079	**0.080**	0.080		0.060	
0.6	0.076	0.076	**0.076**				
0.8	0.068	0.068	0.068		0.061		0.052
0.9	0.063	0.063	0.063	0.062	0.060	0.052	0.059

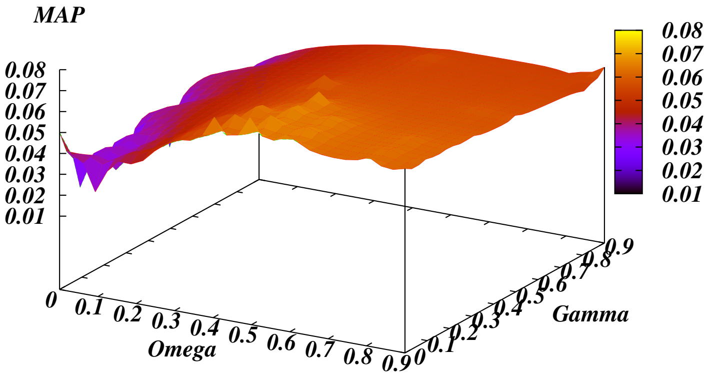

Figure 7.2: Influence of smoothing parameters - odds model

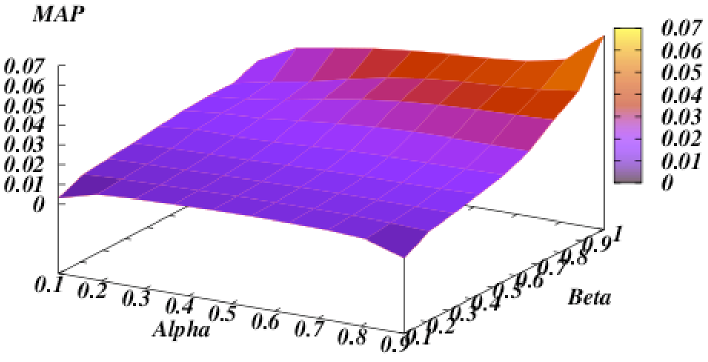

Figure 7.3: Influence of smoothing parameters - prob model

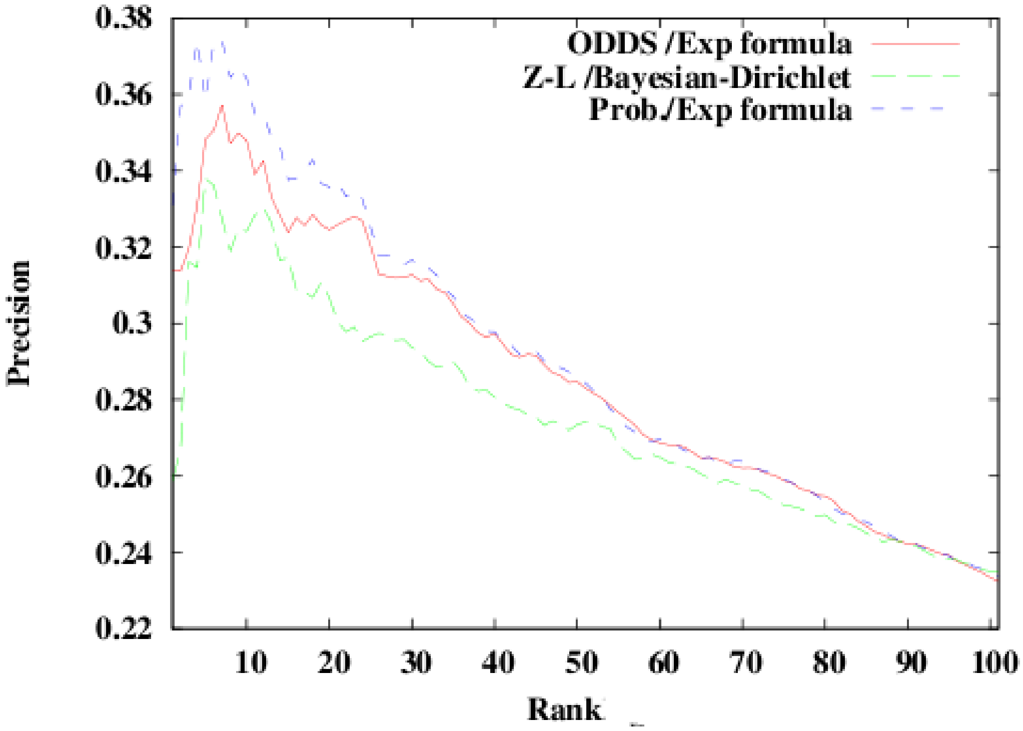

Figure 7.4: Precision-rank curve for the best runs

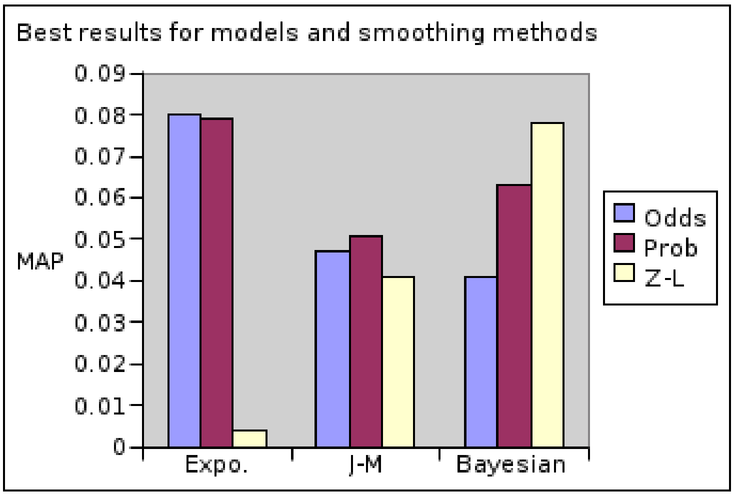

Figure 7.5: Smoothing methods

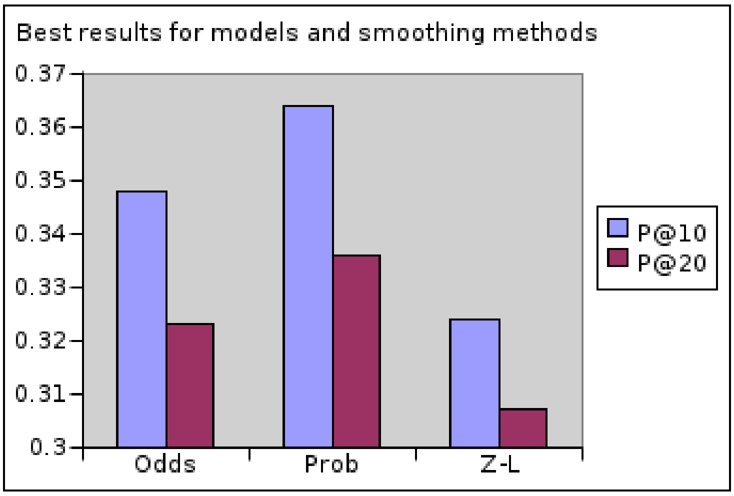

Figure 7.6: Models

Table 7.5: Best results for models and smoothing methods: prob. model ($\alpha = 0.5, \beta = 1$), odds model ($\omega = 0.5, \gamma = 0.2$), Jelink Mercer ($\lambda = 0.7$), Baysian Dirichlet ($\mu = 2,000$), Absolute discount ($\delta = 0.7$)

Smoothing	Model	MAP	Prec at 5	Prec at 10	Prec at 20
Exponential	Odds	**0.080**	**0.348**	**0.348**	**0.323**
	Prob.	**0.079**	**0.359**	**0.364**	**0.336**
	Zhai/Lafferty	0.004	0.015	0.013	0.016
Jelink Mercer	Odds	0.047	0.180	0.180	0.170
	Prob.	0.051	0.180	0.180	0.170
	Zhai/Lafferty	0.040	0.180	0.150	0.140
Baysian Dirichlet	Odds	0.041	0.235	0.235	0.235
	Prob.	**0.063**	**0.300**	**0.290**	**0.290**
	Zhai/Lafferty	**0.078**	**0.338**	**0.324**	**0.307**
Absolute-discount	Odds	0.002	0.009	0.006	0.004
	Prob.	0.002	0.012	0.009	0.007
	Zhai/Lafferty	0.004	0.006	0.006	0.006

7.7 Document length

So far, we have assumed that $P(d)$ grows linearly with document length. Now we assume that $P(d)$ is proportional to l^d, where l is the length and d some smoothing parameter, which possibly may have values below or above 1. Our experiments, (table 7.6) show that best results were achieved with $d = 1.1$.

This assumption was used also by [Abolhassani and Fuhr (2004)], see section 6.3.

Table 7.6: Best results for prob and odds models with dl and dl^i variations: prob. model ($\alpha = 0.5, \beta = 1$), odds model ($\omega = 0.5, \gamma = 0.2$)

Model	Metric	$dl^{0.1}$	$dl^{0.5}$	$dl^{0.9}$	dl^1	$dl^{1.1}$	$dl^{1.2}$	$dl^{1.5}$	$dl^{1.7}$
Odds	MAP	0.028	0.055	0.074	0.080	**0.107**	0.074	0.065	0.057
	P @ 5				0.348	**0.431**	0.341		
	P @ 10				0.364	**0.425**	0.359		
	P @ 20				0.323	**0.394**	0.319		
Prob	MAP	0.032	0.062	0.078	**0.079**	**0.079**	0.078	0.066	0.058
	P @ 5				0.359	**0.369**	0.355		
	P @ 10				0.364	**0.363**	0.349		
	P @ 20				0.336	**0.342**	0.341		

7.8 Logistic regression

With exponential smoothing, our retrieval function has an exponential form, which makes it suitable for logistic regression. In [Cooper et al. (1992)] logistic regression is used for estimating $P(R|q, d)$. In the following, we first discuss logistic regression in general and then describe its application within our framework.

7.8.1 Overview

In statistics, logistic regression (sometimes called the logistic model or logit model) is used for prediction of the probability of occurrence of an event by fitting data to a logistic curve. It is a generalized linear model used for binomial regression. Binary (or binomial) logistic regression is a form of regression which is used when the dependent is a dichotomy and the independents are of any type. Multinomial logistic regression exists to handle the case of dependents with more than two classes, though it is sometimes used for binary dependents also since it generates somewhat different output, as described below. When multiple classes of the dependent variable can be ranked, then ordinal logistic regression is preferred to multinomial logistic regression. Continuous variables are not used as dependents in logistic regression. Unlike logit regression, there can be only one dependent variable.

Logistic regression can be used to predict a dependent variable on the basis of continuous and/or categorical independents and to determine the percent of variance in the dependent variable explained by the independents; to rank the relative importance of independents; to assess interaction effects; and to understand the impact of covariate control variables. The impact of predictor variables is usually explained in terms of odds ratios.

7.8.2 Applying the logistic regression in our model

Following the general form of logistic regression as introduced by [Cooper et al. (1992)], we have independent variables (features) $\overrightarrow{x}$, the relevance judgments are always 0 or 1. Now we estimate the coefficients using logistic regression, instead of using the coefficients we got from our best results.

To get the desired function, we have to compute optimal values for the parameter vector $\overrightarrow{b} = (b_0, b_1, ...b_n)$

In logistic regression, the estimate of $P(R|(\overrightarrow{x}))$ is derived with the formula (7.18)

$$P(R|\overrightarrow{x}) \approx \frac{\mathrm{e}^{b_0+b_1x_1+b_2x_2+....+b_nx_n}}{1+\mathrm{e}^{b_0+b_1x_1+b_2x_2+....+b_nx_n}} \tag{7.18}$$

With this, every possible value lies in the interval [0,1]. The logistic function takes an S-shape (see Figure 7.7), which approximates the value of the class variable y as good as possible for every possible value of $\overrightarrow{x}$, Figure 7.8 shows some exemplary graphs of possible logistic functions. To facilitate the automatic calculation of the above probability estimation formula, it can be converted by introducing the odds notion.

With

$$\mathrm{O}(A) = \frac{P(A)}{P(\bar{A})}$$

of an event A we get

$$\begin{aligned} O(R|\vec{x}) &= \frac{\frac{e^{b_0+b_1x_1+...+b_nx_n}}{1+e^{b_0+b_1x_1+...+b_nx_n}}}{1-\frac{e^{b_0+b_1x_1+...+b_nx_n}}{1+e^{b_0+b_1x_1+...+b_nx_n}}} && (7.19)\\ &= e^{b_0+b_1x_1+...+b_nx_n}. && (7.20)\end{aligned}$$

This can be replaced by so-called Log Odds :
$\log \mathrm{O}(R|\overrightarrow{x}) = b_0 + b_1x_1 + ... + b_nx_n,$

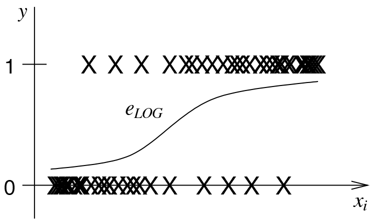

Figure 7.7: Approximation of y with a logistic regression function

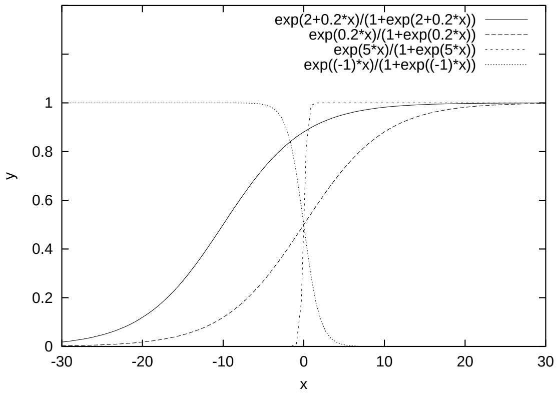

Figure 7.8: Graphs of logistic regression functions

It is enough to calculate the log odds and transform them into the probability estimates only if needed.

To get our desired regression function, we have to compute optimal values for the parameter vector $\overrightarrow{b} = (b_0, b_1, ..., b_n)$. Let

$$a(\overrightarrow{x}, \overrightarrow{b}) = \frac{\mathrm{e}^{\overrightarrow{b}^T \overrightarrow{x}}}{1 + \mathrm{e}^{\overrightarrow{b}^T \overrightarrow{x}}}$$

We further need a learning sample L^{X}, where t is the number of elements in the learning sample. $\mathrm{X} = \{\overrightarrow{x}_1, ..., \overrightarrow{x}_t\}$ is the set of features vectors in the learning sample and $\mathrm{Y} = \{\overrightarrow{y}_1, ..., \overrightarrow{y}_t\}$ the set of the appropriate relevance judgements.

There are two methods for estimating the parameter vector $\overrightarrow{b}$, which we describe in the following sections.

A possible optimization criterion is the least square error estimation. This method assumes that the best-fit curve of a given type is the curve that has the minimal sum of the deviations squared (least square error) from a given set of data. Typically, in tasks of this kind maximum likelihood can also be chosen as an optimization criterion.

In contrast, measuring likelihood aims at maximizing the probability of the observation data. Usually numerical techniques are employed to find the maximum likelihood estimates, where a number of iteration steps has to be performed.

First, the algorithm picks some initial estimates of the parameters. It will compute the likelihood of the data given these parameter estimates. Then it will improve the parameter estimates slightly and recalculate the likelihood of the data. It will do this until it reaches a stop criterion which is usually when the parameter estimates do not change much and sometimes we tell the computer to stop after a certain number of tries or iterations has been reached.

7.8.3 Related work

[Fuhr and Pfeifer (1991)] derived a probabilistic indexing model, which serves as a basis for developing logistic indexing functions. They showed that logistic functions can be applied as indexing functions, and that the definition of description vectors based on the theoretical model is a partially successful strategy. However, additional heuristic strategies such as the development of class-specific functions may yield large improvements. No significant difference were found between logistic and linear (iterated) functions.

[Cooper et al. (1992)] sketched a flexible methodology for designing probabilistic retrieval rules – one that offers the potential of yielding more reliable probability-of-relevance estimates than those attainable by many previous methods, yet not cumbersome at run time. Based on the technique of 'staged logistic regression' on a learning set or test collection, the method exploits a statistical simplifying assumption but corrects for the general upward bias it introduces. The approach is especially appropriate in retrieval environments in which the retrieval clues are grouped or composite, as in the case of subject term matches with several associated properties.

[Cooper et al. (1994)] describes the first experimental application of logistic regression using TREC as testbed. Ray Larson applied the same method to INEX data using the Cheshire system[1]. [Larson (2002)] examines a probabilistic approach to distributed information retrieval using a logistic regression algorithm for estimating collection relevance. The algorithm is compared to other methods for distributed search using test collections developed for distributed search evaluation.

7.8.4 Experiments and results

There are several statistical software packages with logistic regression capabilities, like e.g. SAS, S, SPSS and many others. In our experiments, we used the RapidMiner toolkit.

[1] http://cheshire.berkeley.edu/

Table 7.7: Derived coefficients

Derived type	ω	γ
Grid search	0.500	0.200
LS	0.471	0.283
ML	0.398	0.237

Table 7.8: Results for the odds model using different coefficients

Derived type	MAP	Prec at 5	Prec at 10	Prec at 20
Grid search	0.080	0.348	0.348	0.323
LS	0.067	0.341	0.267	0.230
ML	0.032	0.293	0.262	0.220

The results in tables 7.7 and 7.8 show that the coefficients derived by means of the maximum likelihood (ML) produced poor results. On the other hand the logistic least square method (LS) gave better results and was not far away from our experimental results based on grid search. Maybe the nature of our learning sample lead to these results, where we have considered only the top ranking elements for regression. The regression might need most of the collection to perform better.

Although the coefficients are quite similar, the small variations affects the performance and as depicted in 7.2 and 7.3, the models are very sensitive to these variations.

7.9 Collection effect

INEX is a very special collection, and we want to consider also a standard test collection to measure ad hoc information retrieval effectiveness in the standard way. Comparing results with the known models, we found that our model outperforms the Zhai/Lafferty, DFR and BM25 when using INEX collection. The second part of experiments was conducted on the AP subset of the TREC collection which consists of 240,000 AP documents.

7.9.1 Experiments and results

In our experiments we investigate the performance of the four models: odds model, Zhai/Lafferty, DFR and BM25.

Our odds model was combined with exponential smoothing, the parameters values used are $\gamma = 0.2$ and ω =0.5. The Zhai/Lafferty model was combined with Bayesian Dirichlet smoothing and for the DFR approach, we regarded the best variant only which was DFR_{ab}:

$$w_{ab} = [\frac{1}{tf_2 + 1}][-\log_2 \frac{1}{1+\lambda} - tf_1 \cdot \log_2 \frac{\lambda}{1+\lambda}] \tag{7.21}$$

Regarding the TREC collection, the results show that BM25 is outperforming the odds and ZL models. However the difference between the odds model and ZL model is rather small, and DFR gave the best results. When using the INEX collection, BM25 was very poor, DFR performed well, and odds and ZL models performed best.

For the INEX experiments, a recall-precision graph is shown in Figure 7.9 and precision at ranks 5, 10 and 20 is given in table 7.9.

For the second experiment, a recall-precision graph is shown in Figure 7.10 and precision at ranks 5, 10 and 20 is given in table 7.10.

Table 7.9: MAP, P@5, P@10, P@20 - INEX collection

Method	MAP	Prec at 5	Prec at 10	Prec at 20
Odds	0.080	0.348	0.348	0.323
ZL	0.078	0.338	0.324	0.307
BM25	0.006	0.096	0.087	0.070
DFR w_{ab}	0.044	0.310	0.312	0.285

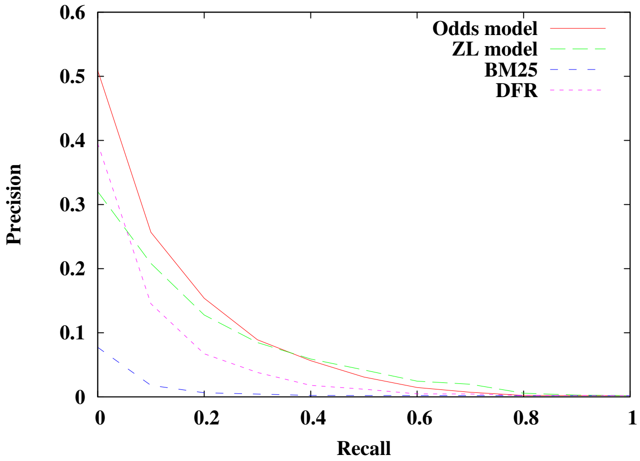

Figure 7.9: Recall-precision curves using INEX collection

Table 7.10: MAP, $P@5$, $P@10$, $P@20$ - TREC collection

Method	MAP	Prec at 5	Prec at 10	Prec at 20
Odds	0.057	0.232	0.211	0.191
ZL	0.061	0.279	0.233	0.228
BM25	0.084	0.445	0.432	0.352
DFR w_{ab}	0.083	0.486	0.435	0.389

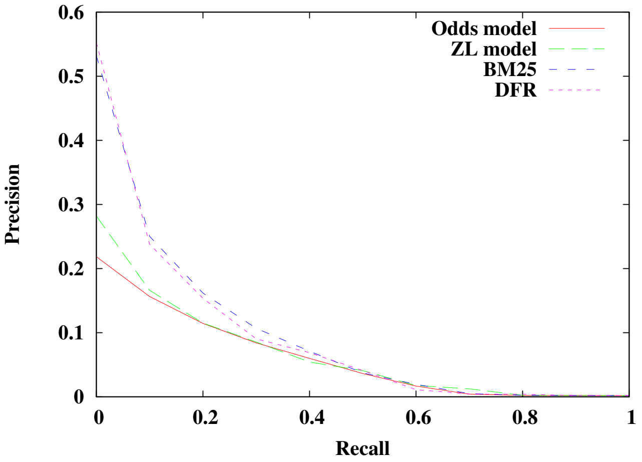

Figure 7.10: Recall-precision curves using TREC collection

Our odds model performed best on the INEX collection, and close to the ZL model on TREC. BM25 shows very poor results using the INEX collection, and DFR was outperforming with the TREC collection and also showed good results using the INEX collection. However, the results indicate that there is no single retrieval approach that can offer optimal performance under all conditions, but that each can offer different advantages in different situations. Performance is dependent to a degree on the type of collection and the set of queries.

7.9.2 Statistical tests

The aim of statistical tests is to know whether or not the difference between two retrieval schemes is really significant or if this difference could have occurred by chance. The statistical significance of results is tested with the paired t-test which calculates the probability that the actual mean difference between the pairs is zero. The t-tests are applied for the measures Mean Average Precision (MAP) and Precision measured at rank 10 (P@10) and at rank 20 (P@20). Table 7.11 shows the results when comparing the Odds model and the Zhai/Lafferty model. The t-test did not show any significant differences, which we believe is due to the small number of queries in the INEX collection. On the other hand, for the TREC collection, we got only one significant difference result showing that the Zhai/Lafferty model is better than the Odds model

Table 7.11: T-test results P(Odds=ZL)

Collection	*Measure*	*P(Odds=ZL)*	*Odds(actual values)*	*ZL(actual values)*
INEX	MAP	0.098	0.080	0.078
	P@10	0.066	0.348	0.324
	P@20	0.088	0.323	0.307
TREC	MAP	0.088	0.057	0.061
	P@10	0.071	0.211	0.233
	P@20	**0.021**	0.191	0.228

Table 7.12: T-test results P(Odds=DFR)

Collection	*Measure*	*P(Odds=DFR)*	*Odds(actual values)*	*DFR(actual values)*
INEX	MAP	**0.024**	0.080	0.044
	P@10	**0.009**	0.348	0.312
	P@20	**0.012**	0.323	0.285
TREC	MAP	**0.029**	0.057	0.083
	P@10	**0.043**	0.211	0.435
	P@20	0.084	0.191	0.389

for P@20.

Table 7.12 shows the significance levels when comparing the Odds model and the DFR model. Using the INEX collection, our tests ensure that the Odds model outperforms the DFR model, since all the measures show significant differences. However with the TREC collection, the results confirm that DFR is outperforming the Odds model - except for the P@20 measure where the difference is not significant.

8 Using the Language Model for XML retrieval

Here we present an extension of the odds language model suitable for structured document retrieval according to the specificity dimension.

8.1 Towards a language model for XML retrieval

Following our language model presented in chapter 7, which can be regarded as a first step of an XML retrieval engine, we discuss here how to extend the odds model to support XML retrieval.

In the next sections, we present the upgraded language model which includes post order and out degree parameters.

8.1.1 Related work on 2-stage retrieval of XML

Fuhr and Großjohann proposed XIRQL [Fuhr and Großjohann (2001)], which combines XQL with probabilistic indexing weights. They model queries as events which are represented in a Boolean algebra. These probabilistic weights are different from those in the language models, as they do not have to add up to 1 across all terms. Augmentation weights are used to consider the hierarchic structure of XML documents. Their system gave good results in the first two INEX rounds.

[Kazai et al. (2001), Kazai et al. (2002)] represent documents as graphs. The document structure is represented using a tree, but horizontal links are allowed among neighbor nodes in the tree. They model nodes in the tree using vectors of term weights. They call combining information in the tree aggregation, and use ordered weighted averaging (OWA) to combine node vectors. OWA is essentially the same as linear interpolation. In [Collins-Thompson et al. (2002)], the authors present the ELIXER query language for XML document retrieval. They adapt XML-QL and WHIRL to allow for similarity matches on document components in the queries. The similarity scores are computed using the cosine similarity on $tf \cdot idf$ weighted vectors representing the query and the document component. Scores for multiple query components are combined by taking the product of the scores. [Myaeng et al. (1998)] represent documents using Bayesian inference networks. The document components act as different document representations, and are combined in the network to produce a structure sensitive score for documents. Only document scores are computed; document components are not ranked. [Hatano et al. (2002)] compute $tf \cdot idf$ vectors for each node in the tree. They compute similarities of text components using cosine similarity, and they use a p-norm function to combine the similarities of the children nodes. The document frequencies are not element specific.

8.1.2 Odds model extension

With structured documents such as XML or HTML, we believe that the information contained in the structure of the document can be used to improve document retrieval. In order to leverage this information, we need to model document structure in the language models. We model structured documents as trees. To index the XML trees we use pre-order and post-order information of the nodes in the XML trees, see section 4.3, for more details.

$$\begin{aligned}\rho_{o,e}(q,d) &= \left(\prod_{t_i \in q^T \cap d^T} P_{ml}(t_i|d)^{\omega} - O(v)\right) \cdot \prod_{t_i \in q^T \cap d^T} P_{avg}(t_i|C)^{-\omega} \\ &\quad \cdot \prod_{t_i \in q^T - d^T} P_{avg}(t_i|C)^{\gamma} \cdot \left(\frac{P(d)}{P(\bar{d})} \cdot \frac{1}{post(v)}\right) \qquad (8.1)\end{aligned}$$

where $O(v)$ is the out-degree of node v and $post(v)$ is its post-order number.

We assume that nodes which have more children give less information for retrieval and therefore we penalize them by subtracting its out-degree. For document length we assume that $P(d)$ is proportional to l^{δ}, where l is the length and δ a smoothing parameter = 1.1 (see section 7.7) which we smooth with the post order number.

We use the same collection as in our previous experiments,(see 5.6.2.3), but this time we regard the specificity dimension only, since specificity targets the most specific answer element in a document. Specificity was measured on a continuous scale with values in [0,1], where 1 represents a fully specific component (i.e. contains only relevant information). We measured retrieval quality with the EPRUM (Expected Precision Recall with User Model) metric (see section 5.5).

Experiments were conducted with the new model to compare it with the odds model. The evaluation results (table 8.1 and figure 8.1) show that the upgraded model improves retrieval effectiveness for XML documents and achieves a significant improvement in the retrieval performance when used for element retrieval and outperforms the odds model which was developed for the standard document retrieval. Especially for the top ranks,

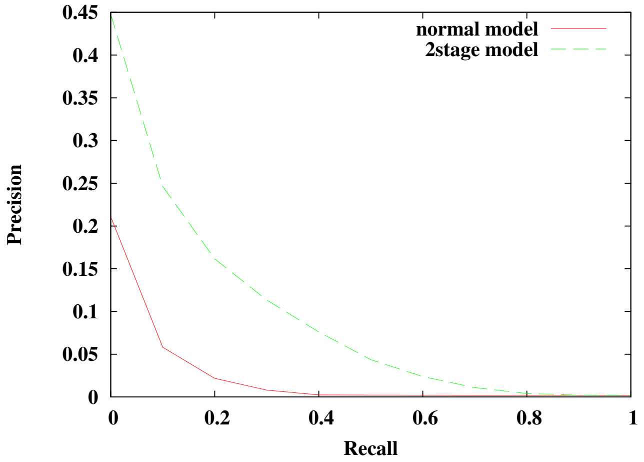

Figure 8.1: Recall-precision curves for retrieval with and without the modified model

Table 8.1: Results before and after modifying the odds model for XML retrieval

Model	MAP	Prec at 5	Prec at 10	Prec at 20
normal model	0.0652	0.203	0.186	0.159
2stage	0.0711	0.372	0.353	0.308

the 2 stage model yields much higher precision values. This also confirms that the odds model is flexible enough to incorporate advanced search techniques.

We compared the "2stage" model with the "mixture" language model from [Sigurbjörnsson et al. (2004)] and the "tree-based" language model from [Ogilvie and Callan (2004)].

In [Sigurbjörnsson et al. (2004)] they estimate their model by taking a linear interpolation of three language models: one for the element itself, one for the article that contains the element, and a third one for the collection. That is, $P(t_i|e)$ is calculated as

$$P(t_i|e) = \lambda_e \cdot P_{ml}(t_i|e) + \lambda_d \cdot P_{ml}(t_i|d) + (1 - \lambda_e - \lambda_d) \cdot P_{ml}(C_i) \tag{8.2}$$

Where $P_{ml}(t_i|e)$ is a language model for element e; $P_{ml}(t_i|d)$ is a language model for document d; and $P_{ml}(C_i)$ is a language model of the collection. The parameters λ_e and λ_d are smoothing parameters. The parameter settings for best runs were $\lambda_e = 0.1$ and λ_d=0.3.

In [Ogilvie and Callan (2004)] they model documents using a tree-based language model. They present a language modeling system for ranking flat text queries against a collection of structured documents. The log of the probability that the document node generated the query is

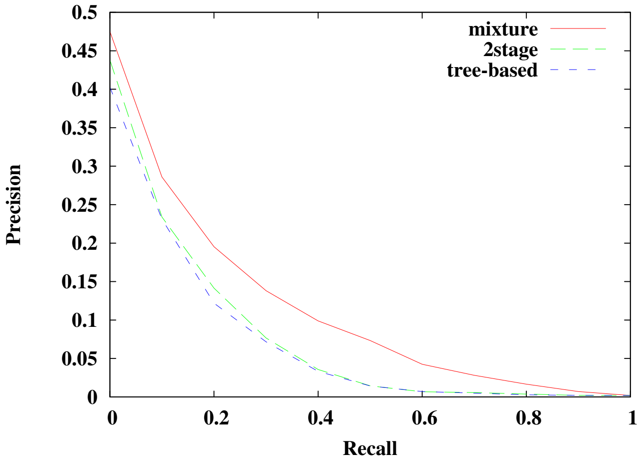

Figure 8.2: Recall-precision curves for three XML retrieval models

Table 8.2: comparison between the three discussed models for XML retrieval

Model	MAP	Prec at 5	Prec at 10	Prec at 20
mixture	0.1074	0.414	0.379	0.336
2stage	0.0711	0.372	0.353	0.308
tree-based	0.0673	0.369	0.345	0.312

$$\log P(q|e) = \sum_{t_i \in q,e} qtf(t) \log \frac{P_{ml}(t_i|e)}{P_{avg}(t_i|C)} + \sum_{t_i \in q} qtf(t) \log P_{avg}(t_i|C) \qquad (8.3)$$

where q is the query, e is the element and $qtf(t)$ is the query term frequency of the term.

The results depicted in figure (8.2) and table (8.2) indicate that the mixture language model shows the best results and the 2stage model outperforms the tree-based model. As can be seen from equation (8.2), the mixture model is the only one that considers the document context of the element to be retrieved. This seems to be the reason for its superior quality. These results also show how language models are flexible and how they can be extended to model and support queries on structured documents.

9 Empirical smoothing

The problem of smoothing is very important in the language modelling approach, where it may significantly affect the performance. The optimal selection of the parameters is usually achieved through experimentation and it varies from application to application and requires extra computation costs. Here we study the connection between smoothing and $tf \cdot idf$ weighting and the role that P_{avg} plays beside smoothing. This leads us to a new smoothing technique called empirical smoothing. [Abdulmutalib and Fuhr (2010)]

9.1 Introduction

Smoothing in language models addresses the problem of data sparsity, where there is rarely enough data to accurately estimate the parameters of a language model. Usually smoothing in language models is used to ensure that none of the words in the collection gets a zero probability.

A connection between smoothing and the $tf \cdot idf$ weighting heuristics appears to be first derived in [Hiemstra and Kraaij (1999)],[Hiemstra (2000)], i.e. $tf \cdot idf$ is not used explicitly in language models, though they use a similar effect. Zhai and Lafferty showed that language modeling contains an *idf*-like component when documents are smoothed with the collection model [Zhai and Lafferty (2001b)].

idf was proposed as a term weighting function by Karen Spärck Jones in 1972 [Jones (1972)], and it has been widely used, usually as part of a $tf \cdot idf$ function. In the context of the entire document set, the importance of a term goes down as the frequency of the term goes up. When the term appears in every document, it is worthless as a differentiator between documents. When the term only appears in a few documents, this will have a high value, i.e. high count, low value; low count, high value.

In language models we have a related effect when we combine the P_{ml} document model with the collection model P_{avg}.

In standard smoothing approaches, it is often claimed that P_{avg} is used only to avoid assigning zero probability to unseen words. Here we start with an analysis of the relationship between P_{ml}, P_{avg} and the probability of relevance of a term. This analysis leads us to a new way of smoothing which we call empirical smoothing. This technique not only solves the problem of unseen word in the document but also improves the accuracy of the estimated language model in general and shows that P_{avg} plays a very similar role as the *idf* weight.

9.2 Empirical Smoothing Technique

As a starting point, we regard the distribution of (P_{ml}, P_{avg}) pairs in relevant and the irrelevant documents and we will show how this relation effects the probability of relevance.

For computing our statistics for a given set of queries, we consider all documents that contain at least one query term, and consider all these query-document pairs. Documents not occurring in the qrel file are implicitly irrelevant. In case the document is relevant, all query terms are regarded as being relevant in this document, otherwise all terms are irrelevant.

Figures 9.1, 9.2 show the frequencies of (P_{ml}, P_{avg}) pairs in relevant/irrelevant query-document pairs. In the relevant documents, low P_{avg} values and high P_{ml} values occur most frequently, whereas the opposite is true for the irrelevant documents.

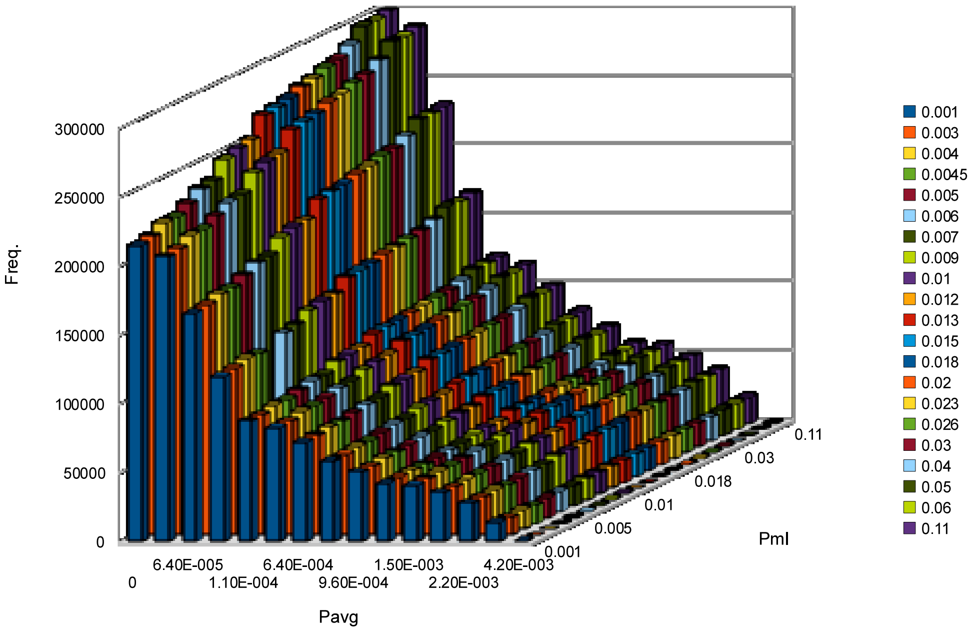

Figure 9.1: Frequency of (P_{ml}, P_{avg}) pairs in relevant documents (INEX collection)

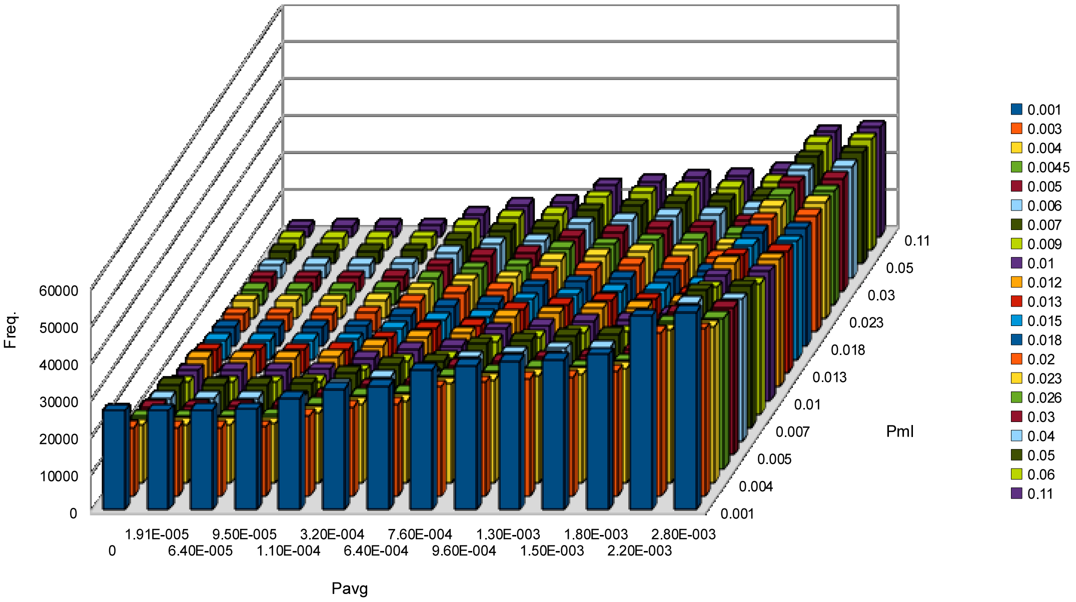

Figure 9.2: Frequency of (P_{ml}, P_{avg}) pairs in irrelevant documents (INEX collection)

However frequency distributions tell only little about relevance. Thus, in the next step, we aimed at relating the (P_{ml}, P_{avg}) pairs of terms to their probability of relevance $P(R|t)$ that a document containing t will be judged relevant to a random query containing t as query term.

For that, we compute the frequency ratio (no. relevant/no. relevant+no. irrelevant) for each 2-dimensional interval of $(P_{ml},\ P_{avg})$ values.

The results are shown in figure 9.3. We can see, that the higher P_{ml} and the smaller P_{avg}, the higher $P(R|t)$.

Most important, we see that P_{avg} is dominating and P_{ml} has only a minor effect. This observation contrasts with the standard justification of smoothing methods in language models, where it is said that P_{ml} is the dominating factor and P_{avg} is used only for dealing with data sparsity. The results also show that for P_{ml}=0 (terms not occurring in the document), $P(R|t)$ is much smaller than for P_{ml}>0. For higher values of P_{avg}, $P(R|t)$ seems to be zero. However, using a logarithmic scale, we can see in figure (9.4) that $P(R|t)$ decreases monotonically as P_{avg} increases.

Finally, in order to verify that these findings are not depending on the collection, the same analysis was conducted with the TREC collection, and the results confirm the same statements, as can be seen from figures 9.5, 9.6. The only difference we can observe from these figures, is that in TREC the P_{avg} slope is not as high as in INEX. One possible reason could be the different definition of relevance, which is less strict in TREC.

In any case, the results for both collections confirm the role of P_{avg} and its effect on the probability of relevance.

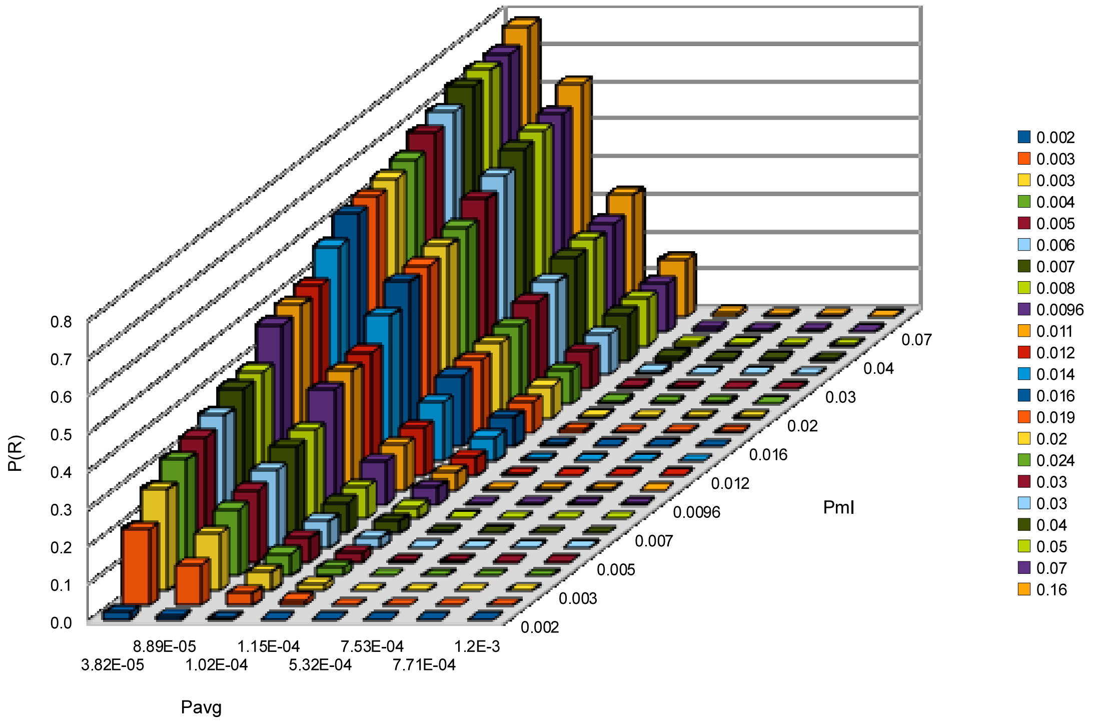

Figure 9.3: $P(R)$ for different (P_{ml}, P_{avg}) values (INEX collection)

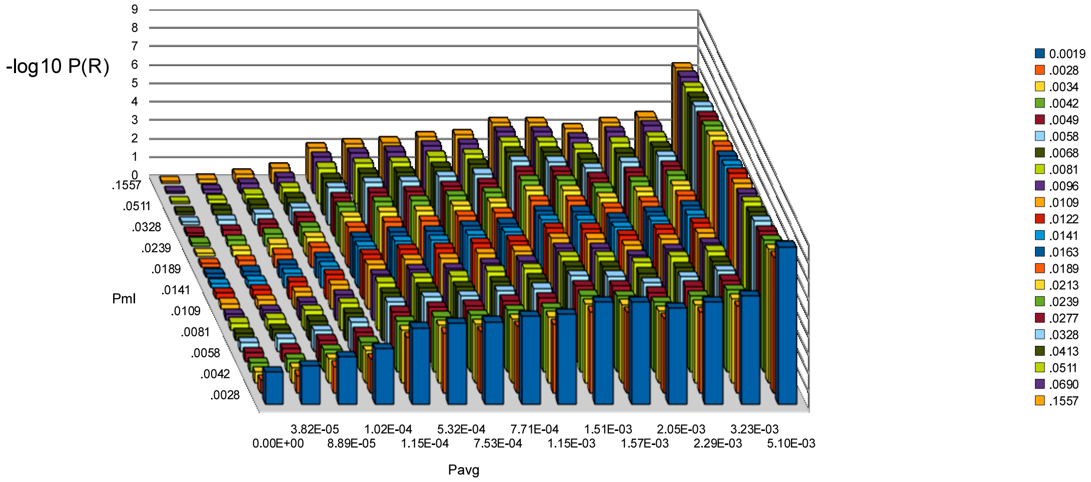

Figure 9.4: $-\log_{10} P(R)$ for different (P_{ml}, P_{avg}) values (INEX collection)

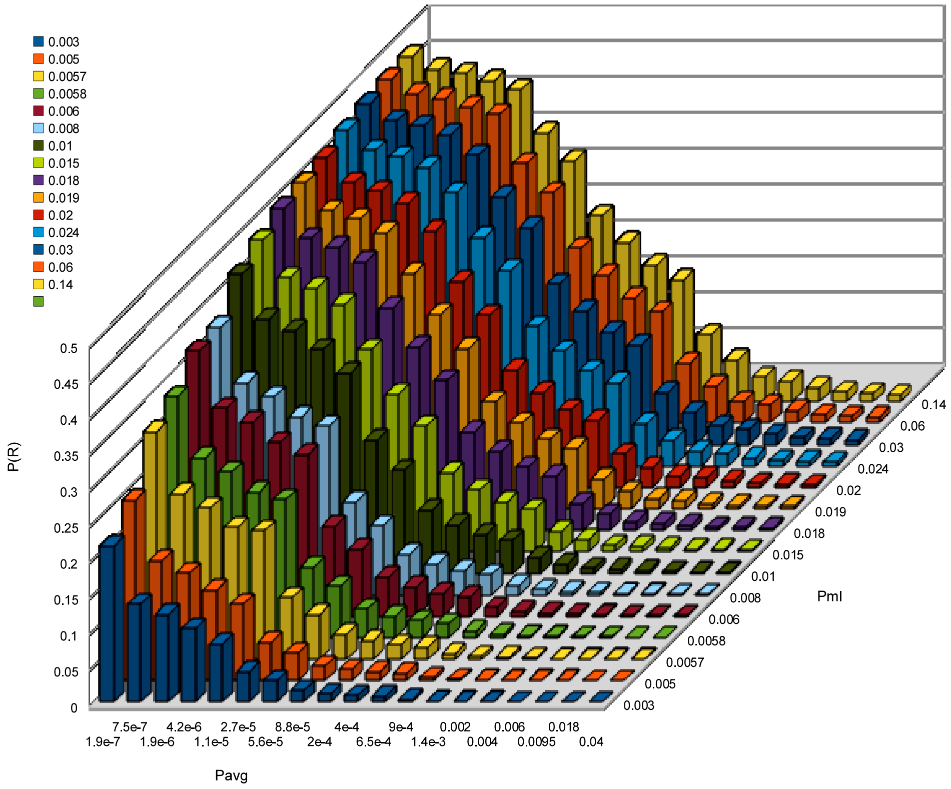

Figure 9.5: $P(R)$ for different (P_{ml}, P_{avg}) values (TREC collection)

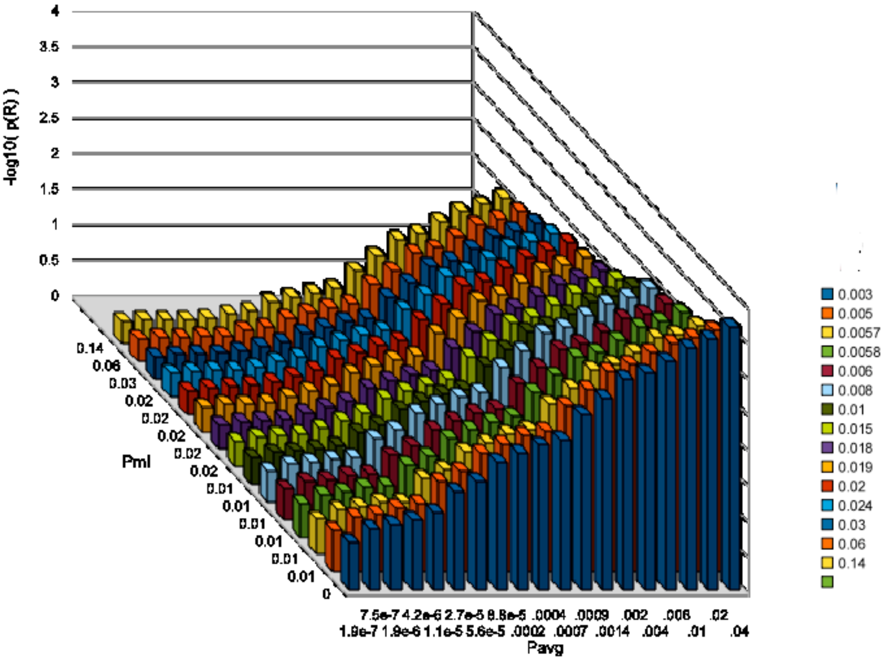

Figure 9.6: $-\log_{10} P(R)$ for different (P_{ml}, P_{avg}) values (TREC collection)

9.3 Implementing empirical smoothing

Based on the observations described above, we now want to present a new approach for smoothing, which we call empirical smoothing. The basic idea is already illustrated in figures 9.3 - 9.6: For each possible combination of (P_{ml}, P_{avg}) values of a term, these plots show the corresponding probability $P(R|t)$. So it seems straight forward to use these values as result of the smoothing process.

9.3.1 Approaches

In principle, there are three different ways for implementing this idea:

- Direct use of interval values:

  As outlined above, we can directly use the probability estimates of $P(R|t)$ from the figures 9.3 - 9.6. Thus, given a (P_{ml}, P_{avg}) pair, we determine the corresponding 2-dimensional interval, and then look up its $P(R|t)$ value from the training set. However, this method needs large amounts of training data to avoid overfitting. Moreover, it does not give us any insights into the relationship between (P_{ml}, P_{avg}) and $P(R|t)$.

- Application of probabilistic classification methods:

  This approach has been investigated already in ([Fuhr (1989b)], [Fuhr and Buckley (1991a)]). As input, the machine learning method would use the raw data underlying figures 9.3 - 9.6, i.e. for each term in each query-document pair considered, we have a training instance consisting of the P_{ml} and P_{avg} values as features and the relevance decision as class variable. In recent years, this kind of approach has also became very popular for developing retrieval functions in the so called 'learning to rank' approaches (see e.g. [Liu (2009)]). Like the previous method, however, this approach operates like a black box, giving us no further insights.

- Application of numeric prediction:

  Here we start with the data shown in figures 9.3 - 9.6, and now seek for a function that describes the relationship between P_{ml}, P_{avg} and $P(R|t)$. As classic smoothing functions perform a similar task, we can compare the outcome of the machine learning method with these functions.

  From these three possibilities, we only consider the last one in the following. Furthermore, we only regard the most simple variant of numeric prediction, namely linear regression.

9.3.2 Linear regression

First, we use a purely linear function of the form:

Table 9.1: Coefficients derived using linear regression

Method	Collection	α	β	δ	γ	Error
LR linear	INEX	0.97	-60.43		0.12	0.053
LR log	INEX	-9.12	-2		9.7	0.011
LR quadratic	INEX	0.97	-209.58	41064.69	0.18	0.022
LR linear cnst.=0	INEX	2.59	-23.4		0	0.060
LR linear	TREC	1.07	-6.93		0.13	0.091
LR log	TREC	-6.23	-0.5		3.43	0.012
LR quadratic	TREC	1.07	-28.03	660.81	0.16	0.041
LR linear cnst.=0	TREC	2.65	-2.69		0	0.094

$$\mathrm{P}_s(t_i|d) = \alpha P_{ml} + \beta P_{avg} + \gamma \tag{9.1}$$

As a second variant, we start from the observation in figure 9.3 that a linear function of P_{avg} may not be very appropriate. Therefore we use $\log(P_{avg})$ instead:

$$\mathrm{P}_s(t_i|d) = \alpha P_{ml} + \beta \log(P_{avg}) + \gamma \tag{9.2}$$

Table 9.1 shows the actual coefficients which have been predicted using linear regression, along with the average squared error. As we can see, replacing (P_{avg}) by its logarithmic improves the error substantially for both collections.

9.3.2.1 Fitting error and the quadratic function

For further analysis, we regard the difference between the linear predictions of equation (9.1) and the actual $P(R|t)$ values, as illustrated in figures 9.7 and 9.8). In the ideal case, there would be random errors; instead, these figures show us systematic deviations from the predicted values. The distribution of these errors suggests that a quadratic function of P_{avg} would be more appropriate:

$$\mathrm{P}_s(t_i|d) = \alpha P_{ml} + \beta P_{avg} + \delta P_{avg}^2 + \gamma \tag{9.3}$$

Looking at the corresponding quadratic errors in table 9.1, we see that the quadratic form is better than the linear one, but not as good as the variant with $\log(P_{avg})$.

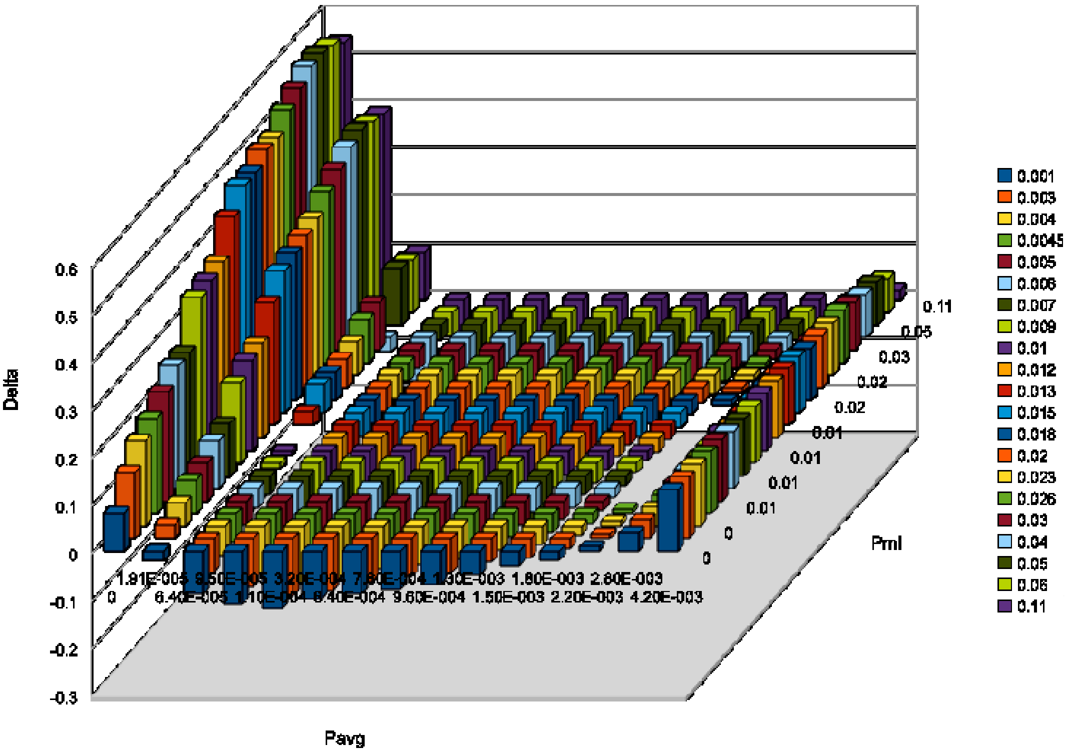

Figure 9.7: Residuals (differences between $P(R)$ and linear regression)(INEX collection)

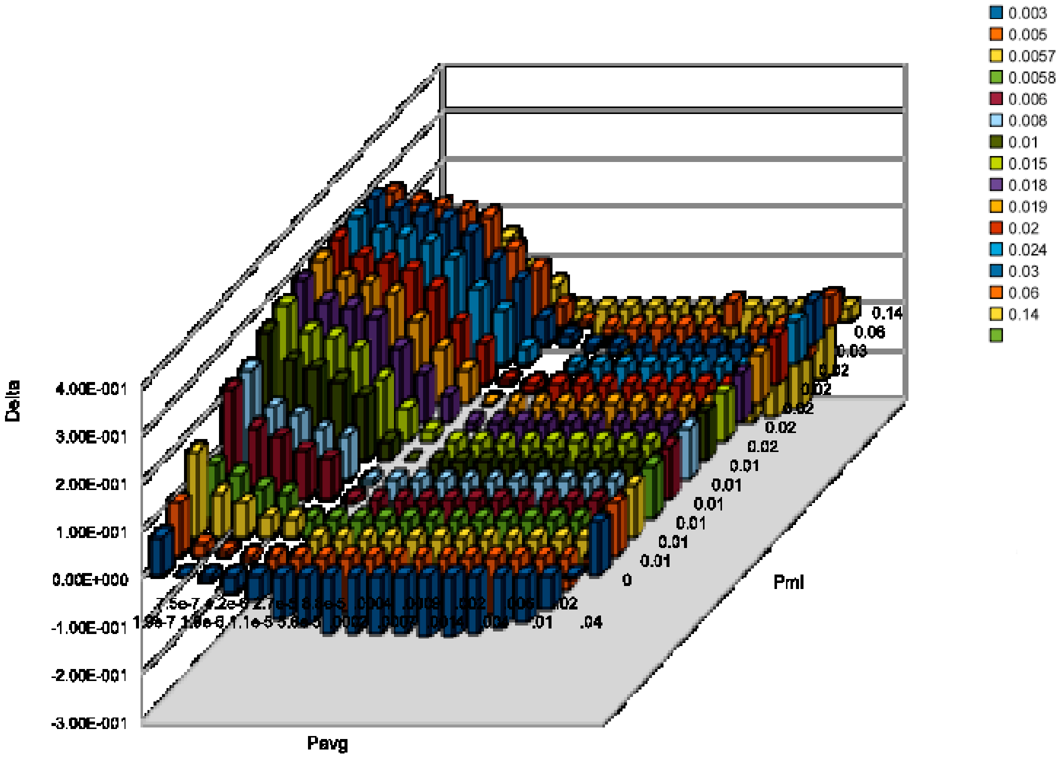

Figure 9.8: Residuals (differences between $P(R)$ and linear regression)(TREC collection)

9.3.2.2 Comparison with Jelinek Mercer smoothing

Since Jelinek Mercer smoothing also uses a linear function with P_{ml} and P_{avg} as inputs, we want to compare its outcome with that of our linear regression. For that, we used the equation 3.15 with $\lambda = 0.7$ which gave the best results for this method. For better comparison, we also tried a variant of equation 9.1, where we dropped the constant γ, so in this case it is very similar to the JM variant. So the structure of the two smoothing functions is the same. However, looking at the corresponding regression coefficients listed in table 9.1, we see that P_{avg} has a negative coefficient, whereas JM smoothing assumes both coefficients to be positive. Moreover, in JM smoothing, P_{ml} is the dominating factor (due to $\lambda = 0.7$), whereas the empirical data as well as the result of our regression put major emphasis on P_{avg}, and P_{ml} just serves as a minor correction factor.

So this comparison does not answer the question why JM smoothing gives fairly reasonable retrieval results, its structure contradicts our empirical findings. A reasonable explanation for this effect remains the subject of further research.

9.3.3 Retrieval experiments

Finally, we performed retrieval experiments with the retrieval function 3.9 and the various smoothing methods. This function is just the basic model, which does not consider document length. Thus, performance can not be optimize, but we are mainly interested in the relative performance of the different smoothing variants. For comparison, we list the results of BM25, ZL and the odds model. The results are depicted in tables 9.2, 9.3 and figures 9.9, 9.10. For the three variants of linear regression, we did not separate between training and test sample, so their results are a bit optimistic. Only for the purely linear form, we performed experiments with 2-fold cross validation, showing that the choice of the training sample has little effect on the quality of results. Comparing the results of the three variants of linear regression, we can see that for both collections, already the linear form gives good results, which can be improved by using one of the variants. For INEX, $\log(P_{avg})$ gives the best quality overall, whereas the quadratic form yields improvements for the top ranking elements only. With TREC, both the logarithmic and the quadratic form are much better than the linear one. In both cases, the quality of JM smoothing is comparable to that of the linear form. With BM25, we confirm the findings from chapter 7: It performs poorly for INEX, but very good for TREC. Overall, these results show that empirical smoothing in combination with nonlinear regression functions is superior to classic smoothing methods.

Table 9.2: MAP, $P@5$, $P@10$, $P@20$ - INEX collection

Method	MAP	Prec at 5	Prec at 10	Prec at 20
LR linear	0.0729	0.355	0.339	0.334
LR log	0.1004	0.397	0.366	0.315
LR quadratic	0.0668	0.389	0.389	0.359
JM	0.0667	0.303	0.245	0.216
LR linear (cv)	0.0862	0.331	0.324	0.299
Odds	0.0800	0.348	0.348	0.323
ZL	0.0780	0.338	0.324	0.307
BM25	0.0063	0.096	0.087	0.070

Table 9.3: MAP, $P@5$, $P@10$, $P@20$ - TREC collection

Method	MAP	Prec at 5	Prec at 10	Prec at 20
LR linear	0.0286	0.283	0.253	0.213
LR log	0.0633	0.359	0.312	0.273
LR quadratic	0.0654	0.304	0.247	0.222
JM	0.0307	0.214	0.238	0.231
LR linear (cv)	0.0355	0.345	0.339	0.333
Odds	0.0572	0.232	0.211	0.191
ZL	0.0611	0.279	0.233	0.228
BM25	0.0844	0.445	0.432	0.352

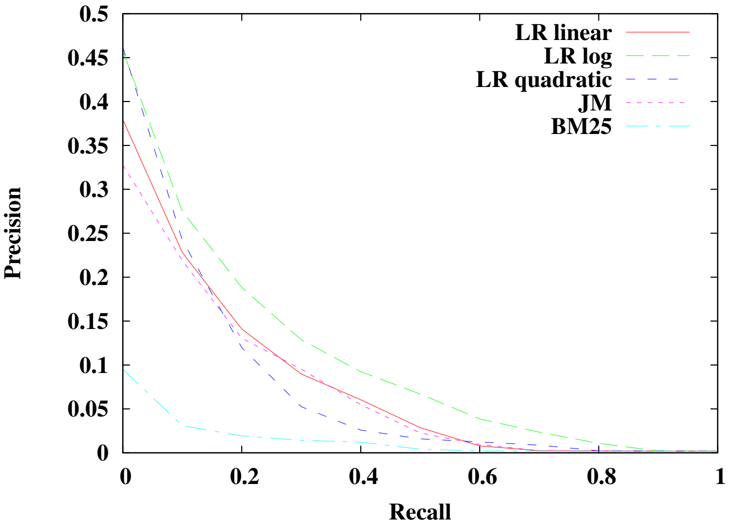

Figure 9.9: Recall-precision curve for various smoothing methods and BM25 (INEX collection)

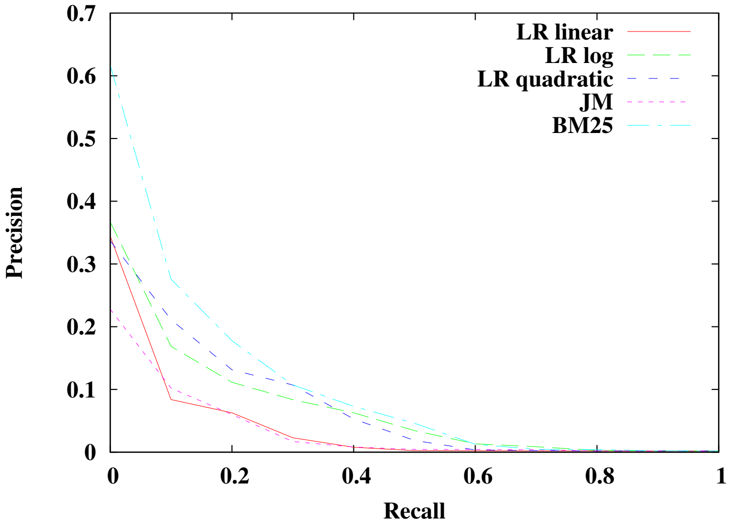

Figure 9.10: Recall-precision curve for various smoothing methods and BM25 (TREC collection)

10 Conclusion and Outlook

Traditionally the issue of term weighting has been addressed mostly in a heuristic way. Language models however can incorporate term frequencies and document length into a probabilistic model. Also, these language models can often be more easily adapted to model various kinds of complex and special retrieval problems than traditional models. Although the language modeling approaches are quite promising and have a great potential for further development, it is an open question whether they will eventually replace the traditional retrieval models.

In this thesis, we first presented a new language model based on an odds formula, as well as a new smoothing method called exponential smoothing. Experiments performed on a collection with large variations in document length showed that document length is an important factor for language models and has a strong effect on the retrieval effectiveness. Models ignoring this parameter lead to very poor results. Our new model along with the new smoothing method showed good results compared to well-known models.

The investigated models as language models in general have been found very sensitive to parameter setting. With variants of the document length parameter and (possibly) document-specific smoothing, there are still possibilities for further improvement. Parameter tuning is generally difficult due to the fact that optimal settings for parameters are often collection/query-dependent and the parameters may interact with others in a complicated way. We applied logistic regression for this purpose, but with limited success.

As a second stage we proposed a specific model for performing XML retrieval, where XML retrieval aims to implement focused retrieval strategies targeting at returning document components, i.e. XML elements in response to a user query. For this purpose, we extended the odds model to incorporate XML-specific features; for that, it takes into account some parameters from the document structure.

To compare the odds model with the well-known BM25 formula, we performed experiments both on the INEX collection as well as on a subset of the TREC collection. Whereas BM25 was clearly inferior on INEX, it showed its superiority on TREC. These results indicate that there is no single retrieval approach that can offer optimal performance under all conditions, but that each one can offer advantages in different situations. Performance is dependent to a degree on the type of collection and the set of queries.

Finally, we have studied the relationship between *idf* weights, the probability of relevance and the language model parameters P_{ml} and P_{avg}. By computing statistics about the distribution of these two parameters in relevant and nonrelevant query-document pairs, we observed a negative correlation between P_{avg} and relevance – confirming the standard approach of *idf* weighting, where rare terms receive higher weight, since they occur more often in relevant documents. This contrasts with language models, where P_{ml} is 'smoothed' by P_{avg}, thus assigning higher weights for higher values of both of these parameters.

Based on the observed negative correlation, we introduced a new smoothing technique called empirical smoothing. As a specific variant, we considered numeric prediction for estimating the probability of relevance $P(R|t,d)$ that a document containing term t with parameters P_{ml} and P_{avg} will be relevant to a query containing term t. As prediction methods, different variants of linear regression were regarded, which led to partially improved retrieval quality. Overall, empirical smoothing seems to be a promising approach, but needs further research.

In future work, other kinds of regression functions (like e.g. logistic regression) should be investigated. Also, the theoretic relationship between *idf* weighting and language models needs more research.

In this thesis, we have performed experiments with two types of test collections with very different characteristics, showing that the collection has a strong influence on the relative performance of different retrieval models. Thus, we need more research with quite diverse collections in order to learn about the factors that influence the performance of retrieval functions. The ultimate goal should be the development of methods that are able to predict retrieval performance for a collection characterized by a set of parameters, without having to perform actual retrieval experiments.

Appendix A

Inex Documents

The next pages contains an example which shows the structure of one of the documents from the IEEE 2005 collection:

Document view: so/2001/s6017

```
<?xml version="1.0"?>
<!DOCTYPE article SYSTEM "/internal/b/projects/inex/2004/inex/dtd
/xmlarticle.dtd">
<article>
  <fno>S6017</fno>
  <doi>10.1041/S6017s-2001</doi>
  <fm>
    <hdr>
      <hdr1>
        <ti>IEEE SOFTWARE</ti>
        <crt>
          <issn>0740-7459</issn>
          /01/$10.00
          <cci>
            <onm>&copy; 2001 IEEE</onm>
          </cci>
        </crt>
      </hdr1>
      <hdr2>
        <obi>
          <volno>Vol. 18</volno>
          <issno>No. 6</issno>
        </obi>
        <pdt>
          <mo>NOVEMBER/DECEMBER</mo>
          <yr>2001</yr>
        </pdt>
        <pp>pp. 17-18</pp>
      </hdr2>
    </hdr>
    <tig>
      <atl>
        Guest Editor’s Introduction: Reports from the Field&mdash;Using
        Extreme Programming and Other Experiences
      </atl>
      <pn>pp. 17-18</pn>
    </tig>
    <au sequence="first">
      <fnm>Wolfgang</fnm>
      <snm>
        <ref type="prb" aid="s6017a1">Strigel</ref>
      </snm>
```

```
        <aff>
          <onm>Software Productivity Center</onm>
        </aff>
      </au>
      <fig>
        <art file="s6017x1.gif"  w="600" h="283" tw="150" th="71"/>
      </fig>
    </fm>
    <bdy>
      <sec>
        <st/>
        <ip1>
          Learning from the successes and failures of others is a quick
          way  to   learn and enlarge our horizon.  Our own experience
          can only cover a narrow path  though the wealth of  existing
           knowledge. Last June, the
          <it>IEEE Software</it>
          Editorial Board decided to make more room for experience reports
           and  give  our  readers  a  forum to  share their own  learning
           experiences with thers.
           If   you  are i nterested in  submitting an experience report,
           please refer to
          <url>www.computer.org/software/genres.htm</url>
          for author guidelines.
        </ip1>
        <p>
          By lucky  coincidence, we had a large backlog of experience
          reports  and  were able to  include  six of  them in this issue.
          On an ongoing basis, we hope to publish two or three shorter
          experience  reports  per issue.    I think you’ll enjoy these
          interesting stories that are typical of the challenges we all
          face in this industry.
          Even if you  were to pick only  one gem from the experience of
          others, it might help you, your project, and your company.
        </p>
        <p>
         The first four articles address the topic of Extreme Programming;
          the  final  two  address a  different set of  experiences  from
          the field.
        </p>
      </sec>
      <sec>
        <st>EXTREME PROGRAMMING IN THE REAL WORLD</st>
        <p>
          Many methodologies have come and gone. Only time will tell if
        one  of  the more  recent methodology innovations, Extreme
        Programming,  will have a  lasting impact on our way to build
        software systems.  Like other  methodologies, XP is not the
        ultimate silver bullet that offers an answer to all development
        problems. But it  has gained  significant momentum and an
        increasing number of software teams are ready to give it a try.
        Our  first  article is not really an  experience  report but an
        interesting  comparison of XP with the more established
        Capability  Maturity Model. As one of the foremost experts on
        CMM,  Mark  Paulk  offers an  opinion on XP as a lightweight
        methodology from the perspective of the heavyweight CMM.
        From my perspective, the difference is not so much the "weight"
        of the methodology than the way they are introduced in an
        organization. XP tends to be a grassroots methodology.
        Developers and development teams typically drive its
        introduction.
        This becomes quite clear from reading the subsequent experience
        reports. CMM, on the other  hand, is typically introduced at the
        corporate  level and then  deployed to development teams. As
        in past  "methodology wars,"  there are heated debates about
        the pros and cons of  the respective approaches. I agree with
        Paulk  that CMM and XP can be considered complementary.
        To establish lasting success, methodologies need buy-in from
        management as well as from the developers.
      </p>
      <p>
        Martin Fowler offers a few links to further information about XP
        and agile methods in the "
        <ref rid="sbs60171" type="sb">Web Resources</ref>
        " sidebar.
      </p>
    </sec>
    <sec>
      <st>TWO MORE REPORTS</st>
      <p>
        The last two articles in the set cover dissimilar experiences,
         but  they  have  one thing in common:  an account of our
         continuous struggle to make  software development more
         efficient.The first article presents a typical example of
         survival struggles in a  rapidly  growing company and its
         attempts to use process to get development activities
         under control.
         The second article describes a technique, called defect logging
         and defect data analysis, that aims to decrease programmers'
        repetitive errors. The author picked one element of the Personal
        Software Process and made it easier to apply.
      </p>
    </sec>
  </bdy>
  <bm>
    <vt id="s6017a1">
      <fig>
        <art file="s6017a1.gif"  w="119" h="143" tw="150" th="180"/>
      </fig>
      <p>
        <b>Wolfgang Strigel</b>
        is the founder and president of Software Productivity Center, a
        consulting and products company, and of QA Labs, a contract
        testing company. His interests include  collaborative software
        development, process improvement, project estimation, testing,
        and   software e ngineering   economics.   He  has a  BSc in
        mathematics from the Technical University, Munich, Germany,
        an MSc in computer science from McGill University, and an MBA
        from Simon Fraser University. Contact him at strigel@spc.ca.
      </p>
    </vt>
    <app id="sbs60171">
      <apt>Extreme Programming and Agile Methods: Web Resources</apt>
      <au sequence="additional">
        <fnm>Martin</fnm>
        <snm>Fowler</snm>
        <aff>
          <onm>ThoughtWorks</onm>
        </aff>
      </au>
      <list type="plain">
        <item>
          <label>
          </label>
          <p>
            <b>Short introduction to XP:</b>
          </p>
        </item>
        <item>
          <label>
          </label>
          <p>
            <url>www.cutter.com/ead/ead0002.html</url>
          </p>
        </item>
        <item>
          <label>
          </label>
          <p>
            <url>
              www.rolemodelsoft.com/articles/xpCorner/xpDistilled.htm
            </url>
          </p>
        </item>
      </list>
      <list type="plain">
        <item>
          <label>
          </label>
          <p>
            <b>XP portals:</b>
          </p>
        </item>
        <item>
          <label>
          </label>
          <p>
            <url>www.xprogramming.com</url>
          </p>
        </item>
        <item>
          <label>
          </label>
          <p>
            <url>www.extremeprogramming.org</url>
          </p>
        </item>
        <item>
          <label>
          </label>
          <p>
            <url>www.rolemodelsoft.com/xp/index.htm</url>
          </p>
        </item>
        <item>
          <label>
          </label>
          <p>
            <url>www.jera.com/techinfo/xpfaq.html</url>
          </p>
        </item>
      </list>
      <list type="plain">
        <item>
          <label>
          </label>
          <p>
            <b>XP mailing lists:</b>
          </p>
        </item>
        <item>
          <label>
          </label>
          <p>
            <url>http://groups.yahoo.com/group/extremeprogramming/</url>
          </p>
        </item>
        <item>
          <label>
          </label>
          <p>
            <url>news:comp.software.extreme-programming</url>
          </p>
        </item>
      </list>
      <list type="plain">
        <item>
          <label>
          </label>
          <p>
            <b>Introduction to agile methods:</b>
          </p>
        </item>
        <item>
          <label>
          </label>
          <p>
          <url>http://martinfowler.com/articles/newMethodology.html</url>
          </p>
        </item>
      </list>
      <list type="plain">
        <item>
          <label>
          </label>
          <p>(All URLs current 15 Oct. 2001)</p>
        </item>
      </list>
    </app>
  </bm>
</article>
```

Bibliography

Abdulmutalib, N. and Fuhr, N. (2008). Language models and smoothing methods for collections with large variation in document length. In Tjoa, A. M. and Wagner, R. R., editors, *DEXA Workshops*, pages 9–14. IEEE Computer Society.

Abdulmutalib, N. and Fuhr, N. (2010). Language models, smoothing, and idf weighting. In *Proc. of the "Information Retrieval 2010" Workshop at LWA 2010, Kassel, Germany*, pages 169–174.

Abney, S. and Light, M. (1999). Hiding a semantic hierarchy in a markov model. In *In Proceedings of the Workshop on Unsupervised Learning in Natural Language Processing, ACL*, pages 1–8.

Abolhassani, M. and Fuhr, N. (2004). Applying the divergence from randomness approach for content-only search in XML documents. In McDonald, S. and Tait, J., editors, *26th European Conference on Information Retrieval Research (ECIR 2004)*, pages 409–419, Heidelberg et al. Springer.

Abolhassani, M., Fuhr, N., Gövert, N., and Großjohann, K. (2002). HyREX: Hypermedia retrieval engine for XML. Research report, University of Dortmund, Department of Computer Science, Dortmund, Germany.

Amati, G. (2003). *Probability Models for Information Retrieval based on Divergence from Randomness.* PhD thesis, University of Glasgow.

Amati, G. and van Rijsbergen, C. (2002a). Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness. *ACM Trancactions on Information Systems*, 20(4):357–389.

Amati, G. and van Rijsbergen, C. J. (2002b). Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389.

Baldi, Frasconi, P., and Smyth, P. (1997). *Modelling the Internet and the Web - Probabilistic Methods and Algorithms,.* Wiley & Sons.

Berger, A. and Lafferty, J. (1999). Information retrieval as statistical translation. pages 222–229, New York. ACM.

Biebricher, P., Fuhr, N., Knorz, G., Lustig, G., and Schwantner, M. (1988). The automatic indexing system AIR/PHYS - from research to application. In *11th International Conference on Research and Development in Information Retrieval*, pages 333–342, Grenoble, France. Presses Universitaires de Grenoble.

Blanken, H. M., Grabs, T., Schek, H.-J., Schenkel, R., and Weikum, G., editors (2003). *Intelligent Search on XML Data. Applications, Languages, Models, Implementations, and Benchmarks*, volume 2818 of *Lecture Notes in Computer Science*. Springer, Heidelberg et al.

Bookstein, A. and Swanson, D. R. (1975). A decision theoretic foundation for indexing. *Journal of the American Society for Information Science*, 26:45–50.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In *Proc. of WWW, pages 107–117, Brisbane, Australia*. Elsevier Science.

Broglio, J., Callan, J. P., Croft, W. B., and Nachbar, D. W. (1995). Document retrieval and routing using the inquery system. In Harman, D., editor, *In Proceeding of Third Text Retrieval Conference (TREC-3)*, pages 29–38, Gaithersburg, Md. 20899. National Institute of Standards and Technology.

Brown, P. F., Cocke, J., Pietra, S. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Burger, J. D., Palmer, D., and Hirschman, L. (1998). Named entity scoring for speech input. In *Proceedings of the 17th international conference on Computational linguistics*, pages 201–205, Morristown, NJ, USA. Association for Computational Linguistics.

Chowdhury, A., McCabe, M. C., Grossman, D., and Frieder, O. (2002). Document normalization revisited. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 381–382, New York, NY, USA. ACM.

Collins-Thompson, K., Ogilvie, P., Zhang, Y., and Callan, J. (2002). Information filtering, novelty detection, and named-page finding. In *In Proceedings of the 11th Text Retrieval Conference*.

Cooper, W. S. (1991). Some inconsistencies and misnomers in probabilistic IR. In *Proceedings of the Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 57–61, New York. ACM.

Cooper, W. S. (1995). Some inconsistencies and misidentified modeling assumptions in probabilistic information retrieval. *ACM Transactions on Information Systems*, 13(1):100–111.

Cooper, W. S., Chen, A., and Gey, F. C. (1994). Full text retrieval based on probabilistic equations with coefficients fitted by logistic regression. In Harman, D., editor, *The*

Second Text REtrieval Conference (TREC-2), pages 57–66, Gaithersburg, Md. 20899. National Institute of Standards and Technology.

Cooper, W. S., Gey, F. C., and Dabney, D. P. (1992). Probabilistic retrieval based on staged logistic regression. In Belkin, N. J., Ingwersen, P., and Pejtersen, A. M., editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992*, pages 198–210, New York. ACM.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory.* John Wiley & Sons, New York, NY, USA.

Croft, W. B., Harper, D., Kraft, D. H., and Zobel, J., editors (2001). *Proceedings of the 24th Annual International Conference on Research and development in Information Retrieval*, New York. ACM.

Croft, W. B., Moffat, A., van Rijsbergen, C. J., Wilkinson, R., and Zobel, J., editors (1998). *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York. ACM.

Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. (1992). A practical part-of-speech tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 133–140, Morristown, NJ, USA. Association for Computational Linguistics.

Frommholz, I. (2008). *A Probabilistic Framework for Information Modelling and Retrieval Based on User Annotations on Digital Objects.* PhD thesis, University of Duisburg-Essen. Submitted.

Fuhr, N. (1989a). Models for retrieval with probabilistic indexing. *Information Processing and Management*, 25(1):55–72.

Fuhr, N. (1989b). Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems*, 7(3):183–204.

Fuhr, N. (2000). Probabilistic Datalog: Implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51(2):95–110.

Fuhr, N. (2001a). Language models and uncertain inference in information retrieval. In *Proc. Workshop on Language Modelling and Information Retrieval*, pages 6–11, Pittsburgh, PA. Carnegie Mellon University.

Fuhr, N. (2001b). Models in information retrieval. In Agosti, M., Crestani, F., and Pasi, G., editors, *Lectures in Information Retrieval*, pages 21–50. Springer, Heidelberg et al.

Fuhr, N. and Buckley, C. (1991a). A Probabilistic Learning Approach for Document Indexing. *ACM Trancactions on Information Systems*, 9(3):223–248.

Fuhr, N. and Buckley, C. (1991b). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9(3):223–248.

Fuhr, N., Gövert, N., Kazai, G., and Lalmas, M., editors (2003). *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the First INEX Workshop. Dagstuhl, Germany, December 8–11, 2002*, ERCIM Workshop Proceedings, Sophia Antipolis, France. ERCIM. `http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf`.

Fuhr, N. and Großjohann, K. (2001). XIRQL: A query language for information retrieval in XML documents. In Croft et al. (2001), pages 172–180.

Fuhr, N. and Großjohann, K. (2002). XIRQL: An XML query language based on information retrieval concepts. (Submitted for publication).

Fuhr, N. and Lalmas, M. (2007). Advances in xml retrieval: the inex initiative. In *IWRIDL '06: Proceedings of the 2006 international workshop on Research issues in digital libraries*, pages 1–6, New York, NY, USA. ACM.

Fuhr, N., Lalmas, M., and Malik, S., editors (2004). *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop. Dagstuhl, Germany, December 15–17, 2003*. `http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf`.

Fuhr, N., Lalmas, M., Malik, S., and Kazai, G., editors (2006a). *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005, Revised Selected Papers*, volume 3977 of *LNCS*. Springer.

Fuhr, N., Lalmas, M., Malik, S., and Kazai, G., editors (2006b). *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005), Dagstuhl 28-30 November 2005, Lecture Notes in Computer Science*, volume 3977. Springer-Verlag GmbH.

Fuhr, N., Lalmas, M., Malik, S., and Szlavik, Z., editors (2005). *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers*, volume 3493. Springer-Verlag GmbH. http://www.springeronline.com/3-540-26166-4.

Fuhr, N. and Pfeifer, U. (1991). Combining model-oriented and description-oriented approaches for probabilistic indexing. In *Proceedings of the Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 46–56, New York. ACM.

Fuhr, N. and Rölleke, T. (1998). HySpirit – a probabilistic inference engine for hypermedia retrieval in large databases. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT)*, pages 24–38, Heidelberg et al. Springer.

Gövert, N., Fuhr, N., Abolhassani, M., and Großjohann, K. (2003a). Content-oriented XML retrieval with HyREX. In Fuhr et al. (2003), pages 26–32. http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf.

Gövert, N. and Kazai, G. (2003). Overview of the INitiative for the Evaluation of XML retrieval (INEX) 2002. In Fuhr et al. (2003), pages 1–17. http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf.

Gövert, N., Kazai, G., Fuhr, N., and Lalmas, M. (2003b). Evaluating the effectiveness of content-oriented XML retrieval. Technical report, University of Dortmund, Computer Science 6.

Greiff, W. (2001). Is it the language model in language modeling? In *Proceedings of the Workshop on Language Modeling and Information Retrieval*.

Greiff, W., Morgan, A., Fish, R., Richards, M., and Kundu, A. (2001). Fine-grained hidden markov modeling for broadcast-news story segmentation. In *HLT '01: Proceedings of the first international conference on Human language technology research*, pages 1–5, Morristown, NJ, USA. Association for Computational Linguistics.

Harman, D. (1995). Overview of the second text retrieval conference (TREC-2). *Information Processing and Management*, 31(03):271–290.

Harter, S. D. (1975a). A probabilistic approach to automatic keyword indexing. part I: On the distribution of speciality words in a technical literature. *Journal of the American Society for Information Science*, 26:197–206.

Harter, S. D. (1975b). A probabilistic approach to automatic keyword indexing. part II: An algorithm for probabilistic indexing. *Journal of the American Society for Information Science*, 26:280–289.

Hatano, K., Kinutani, H., Yoshikawa, M., and Uemura, S. (2002). Information retrieval system for xml documents. In Hameurlain, A., Cicchetti, R., and Traunmüller, R., editors, *DEXA*, volume 2453 of *Lecture Notes in Computer Science*, pages 758–767. Springer.

Hawking, D. and Craswell, N. (2005). Very large scale retrieval and web search. In Voorhees, E. and Harman, D., editors, *Chapter in TREC: Experiment and Evaluation in Information Retrieval.* MIT Press. http://es.csiro.au/pubs/trecbook_for_website.pdf (ISBN 0262220733).

Hawking, D., Craswell, N., Bailey, P., and Griffiths, K. (2001). Measuring search engine quality. *Inf. Retr.*, 4(1):33–59.

Hiemstra, D. (1998). A linguistically motivated probabilistic model of information retrieval. In *Lecture Notes In Computer Science - Research and Advanced Technology for Digital Libraries - Proceedings of the second European Conference on Research and Advanced Technology for Digital Libraries: ECDL'98*, pages 569–584. Springer Verlag.

Hiemstra, D. (2000). A probabilistic justification for using tf idf term weighting in information retrieval. *International Journal on Digital Libraries*, 3(2):131–139.

Hiemstra, D. (2001). *Using Language Models for Information Retrieval.* PhD thesis, University of Twente, Enschede.

Hiemstra, D. and de Vries, A. (2000). Relating the new language models of information retrieval to the traditional retrieval models. Technical Report TR-CTIT-00-09, CTIT, University of Twente.

Hiemstra, D. and Kraaij, W. (1999). Twenty-one at trec-7: Ad-hoc and cross-language track. In *In Proc. of Seventh Text REtrieval Conference (TREC-7)*, pages 227–238.

Järvelin, K., Allen, J., Bruza, P., and Sanderson, M., editors (2004). *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York. ACM.

Järvelin, K., Beaulieu, M., Baeza-Yates, R., and Myaeng, S. H., editors (2002). *Proceedings of the 25th Annual International Conference on Research and Development in Information Retrieval*, New York. ACM.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Jelinek, F. (1998). *Statistical Methods for Speech Recognition,*. MIT Press.

Jelinek, F. and Mercer, R. L. (1980). Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice.*

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.

Jones, K. S., Robertson, S. E., Hiemstra, D., and Zaragoza, H. (2003). Language modelling and relevance. In Croft, W. B. and Lafferty, J., editors, *Language Modeling for Information Retrieval*, volume 13 of *The Information Retrieval Series*, pages 57–71. Springer Verlag, Berlin.

Kazai, G. and Lalmas, M. (2005). Inex 2005 evaluation measures. In Fuhr et al. (2006b), pages 16–29.

Kazai, G. and Lalmas, M. (2006). Notes on what to measure in inex. In Fuhr et al. (2006b), pages 22–38.

Kazai, G., Lalmas, M., and de Vries, A. P. (2004). The overlap problem in content-oriented XML retrieval evaluation. In Järvelin et al. (2004), pages 72–79.

Kazai, G., Lalmas, M., and Roelleke, T. (2002). Focussed structured document retrieval. In *In Proceedings of the 9 th Symposium on String Processing and Information Retrieval (SPIRE 2002)*, pages 241–247. Springer.

Kazai, G., Lalmas, M., and Rölleke, T. (2001). A model for the representation and focussed retrieval of structured documents based on fuzzy aggregation. In *Aggregation, 8th International Symposium on String Processing and Information Retrieval (SPIRE2001), pp 123-135, Laguna de*, pages 123–135.

Kraaij, W. (2004). *Variations on language modeling for information retrieval.* PhD thesis, University of Twente, Enschede.

Kwok, K. L. (1990). Experiments with a component theory of probabilistic information retrieval based on single terms as document components. *ACM Transactions on Information Systems*, 8:363–386.

Lafferty, J. and Zhai, C. (2001a). Probabilistic ir models based on document and query generation. In J. Callan, B. C. and Lafferty, J., editors, *Proceedings of workshop on Language Modeling and Information Retrieval.*

Lafferty, J. D. and Zhai, C. (2001b). Document language models, query models, and risk minimization for information retrieval. In Croft et al. (2001), pages 111–119.

Lalmas, M. and Tombros, A. (2007). Inex 2002 - 2006: Understanding xml retrieval evaluation. In *DELOS Conference*, pages 187–196.

Larson, R. R. (2002). A logistic regression approach to distributed ir. In Järvelin et al. (2002), pages 399–400.

Lavrenko, V., Choquette, M., and Croft, W. B. (2002). Cross-lingual relevance models. In Järvelin et al. (2002), pages 175–182.

Lavrenko, V. and Croft, W. B. (2001). Relevance based language models. In Croft et al. (2001), pages 120–127.

Lesk, M. (1995). The seven ages of information retrieval. In *Proc. Conf. 50th Anniversary of As We May Think.* ACM.

Lewis, M. (1998). Designing for human-agent interaction. *AI Magazine*, 1998:67–78.

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331.

Losada, D. E. and Azzopardi, L. (2008). An analysis on document length retrieval trends in language modeling smoothing. *Information Retrieval*, 11(2):109–138.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing.* The MIT Press.

Maron, M. E. and Kuhns, J. L. (1960). On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244.

Mei, Q., Fang, H., and Zhai, C. (2007). A study of poisson query generation model for information retrieval. In Kraaij, W., de Vries, A. P., Clarke, C. L. A., Fuhr, N., and Kando, N., editors, *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 319–326, New York, NY, USA. ACM.

Metzler, D., Lavrenko, V., and Croft, B. W. (2004). Formal multiple-bernoulli models for language modeling. In Järvelin et al. (2004), pages 540–541.

Miller, D. R. H., Leek, T., and Schwartz, R. M. (1999). A hidden markov model information retrieval system. pages 214–221, New York. ACM.

Mitra, M., Buckley, C., Singhal, A., and Cardie, C. (1997). An analysis of statistical and syntactic phrases. In Devroye, L. and Chrisment, C., editors, *RIAO*, pages 200–217.

Myaeng, S. H., Jang, D.-H., Kim, M.-S., and Zhoo, Z.-C. (1998). A flexible model for retrieval of SGML documents. In Croft et al. (1998), pages 138–145.

Nottelmann, H. (2005). PIRE: An extensible IR engine based on probabilistic datalog. In Losada, D. E. and Luna, J. M. F., editors, *27th European Conference on Information Retrieval Research (ECIR 2005)*.

Ogilvie, P. and Callan, J. (2004). Using language models for flat text queries in XML retrieval. In Fuhr et al. (2004), pages 12–18. `http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf`.

Ogilvie, P. and Lalmas, M. (2006). Investigating the exhaustivity dimension in content-oriented xml element retrieval evaluation. In *Proceedings of ACM CIKM*, New York. ACM.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, California.

Piwowarski, B. (2005). Eprum metrics and inex 2005. In Fuhr et al. (2006a), pages 30–42.

Piwowarski, B. (2006). Eprum metrics and inex 2005. In Fuhr, N., Lalmas, M., Malik, S., and Kazai, G., editors, *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005)*. Springer-Verlag.

Piwowarski, B. and Dupret, G. (2006). Evaluation in (xml) information retrieval: expected precision-recall with user modelling (eprum). pages 260–267.

Piwowarski, B. and Gallinari, P. (2004). Expected ratio of relevant units: A measure of structured information retrieval. In Fuhr et al. (2004), pages 158–166. `http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf`.

Ponte, J. and Croft, W. (1998a). 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. ACM Press.

Ponte, J. M. and Croft, W. B. (1998b). A language modeling approach to information retrieval. In Croft et al. (1998), pages 275–281.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14:130–137.

Rabiner, L. R., Levinson, S. E., and Sondhi, M. M. (1983). On the application of vector quantization and hidden Markov models to speaker-independent, isolated word recognition. *Bell System Technical Journal*, 62(4):1075–1105.

Raghavan, V. V., Bollmann, P., and Jung, G. S. (1989). A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems*, 7(3):205–229.

Robertson, S. E. (1977). The probability ranking principle in IR. *Journal of Documentation*, 33:294–304.

Robertson, S. E. and Sparck Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146.

Robertson, S. E., van Rijsbergen, C. J., and Porter, M. F. (1981). Probabilistic models of indexing and searching. In *Information Retrieval Research*, pages 35–56. Butterworths, London.

Robertson, S. E. and Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In Croft, B. W. and van Rijsbergen, C. J., editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, London, et al. Springer-Verlag.

Robertson, S. E., Walker, S., Beaulieu, M. M., Gatford, M., and Payne, A. (1998). Okapi at TREC-7. In *Proceedings of the 7th Text Retrival Converence (TREC-7)*.

Robertson, S. E., Walker, S., Hancock-Beaulieu, M., Gull, A., and Lau, M. (1992). Okapi at TREC. In *Text REtrieval Conference*, pages 21–30.

Robertson, S. E., Walker, S., Jones, S., and Hancock-Beaulieu, M. M. (1995). Okapi at TREC-3. In *Proceedings of the 3rd Text Retrieval Converence (TREC-3)*, pages 109–126, Springfield, Virginia, USA. NTIS.

Rölleke, T. (1998). *POOL: Probabilistic Object-Oriented Logical Representation and Retrieval of Complex Objects*. PhD thesis, University of Dortmund, Germany.

Rosenfeld, R. (2000). Two decades of statistical language modeling: Where do we go from here. In *Proceedings of the IEEE, 88(8)*.

Salton, G. and Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523.

Shannon, C. E. (1951). Prediction and entropy of printed english. *Bell Systems Technical Journal*, 30:50–64.

Sigurbjörnsson, B., Kamps, J., and de Rijke, M. (2004). Mixture models, overlap, and structural hints in xml element retrieval. In Fuhr et al. (2005), pages 196–210. http://www.springeronline.com/3-540-26166-4.

Singhal, A., Buckley, C., and Mitra, M. (1996). Pivoted document length normalization. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–29, New York, NY, USA. ACM.

Song, F. and Croft, W. B. (1999). A general language model for information retrieval. In *CIKM*, pages 316–321.

Sparck Jones, K. and Robertson, S. (2001). Lm vs. pm: where is the relevance? In *Proceedings of the Workshop on Language Modeling and Information Retrieval*.

Spoken, K. N. (1999). A maximum likelihood ratio information retrieval model.

Trotman, A. (2005). Wanted: Element retrieval users. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*.

Trotman, A. and Sigurbjornsson, B. (2005). Narrowed extended XPath I (NEXI). In Fuhr et al. (2005). http://www.springeronline.com/3-540-26166-4.

Turtle, H. and Croft, W. B. (1990). Inference networks for document retrieval. In *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 1–24, New York. ACM.

Turtle, H. and Croft, W. B. (1991). Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222.

Ullman, J. D. (1988). *Principles of Database and Knowledge-Base Systems*, volume I. Computer Science Press, Rockville (Md.).

van Rijsbergen, C. J. (1977). A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, 33:106–119.

van Rijsbergen, C. J. (1986). A non-classical logic for information retrieval. *The Computer Journal*, 29(6):481–485.

Voorhees, E. (2000). Overview of TREC 2001. In Voorhees, E. M. and Harman, D. K., editors, *The Tenth Text REtrieval Conference (TREC 2001)*. NIST, Gaithersburg, MD, USA.

Voorhees, E. (2003). Common evaluation measures. pages 1–13.

Voorhees, E. M. and Buckland, L. P., editors (2005). *The Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, USA. NIST.

Wong, S. K. M. and Yao, Y. Y. (1995). On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):38–68.

Yamron, J. P., Carp, I., Gillick, L., Lowe, S., and van Mulbregt, P. (1998). A hidden markov model approach to text segmentation and event tracking. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 333–336, Seattle, WA. IEEE.

Zhai, C. and Lafferty, J. (2001a). The dual role of smoothing in the language modeling approach. In *Proceedings of the Workshop on Language Models for Information Retrieval (LMIR) 2001*, pages 31–36.

Zhai, C. and Lafferty, J. (2001b). A study of smoothing methods for language models applied to ad hoc information retrieval. In Croft et al. (2001).

Zhai, C. and Lafferty, J. (2002). Two-stage language models for information retrieval. In Järvelin et al. (2002), pages 49–56.