USING LANGUAGE MODELS FOR INFORMATION RETRIEVAL

Djoerd Hiemstra

Neslia Paniculata

USING LANGUAGE MODELS
FOR INFORMATION RETRIEVAL

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. F.A. van Vught,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 19 januari 2001 te 16.45 uur.

door

Djoerd Hiemstra

geboren op 20 februari 1971
te Zwolle

Dit proefschrift is goedgekeurd door de promotor,
prof. dr. F.M.G. de Jong.

One, and two, and three police persons spring out of the shadows

Down the corner comes one more

And we scream into that city night: "three plus one makes four!"

Well, they seem to think we're disturbing the peace

But we won't let them make us sad

'Cause kids like you and me baby, we were born to add

Born To Add, Sesame Street
(sung to the tune of Bruce Springsteen's Born to Run)

to Ursula

Preface

In October 1996, I got a position as a research assistant working on the Twenty-One project. The project aimed at providing a software architecture that supports a multilingual community of people working on local Agenda 21 initiatives in exchanging ideas and publishing their work. Local Agenda 21 initiatives are projects of local governments, aiming at sustainable processes in environmental, human, and economic terms. The projects cover themes like combating poverty, protecting the atmosphere, human health, freshwater resources, waste management, education, etc. Documentation on local Agenda 21 initiatives are usually written in the language of the local government, very much unlike documentation on research in e.g. information retrieval for which English is *the* language of international communication. Automatic cross-language retrieval systems are therefore a helpful tool in the international cooperation between local governments. Looking back, I regret not being more involved in the non-technical aspects of the Twenty-One project. To make up for this loss, many of the examples in this thesis are taken from the project's domain.

Working on the Twenty-One project convinced me that solutions to cross-language information retrieval should explicitly combine translation models and retrieval models into one unifying framework. Working in a language technology group, the use of language models seemed a natural choice. A choice that simplifies things considerably for that matter. The use of language models for information retrieval practically reduces ranking to simply adding the occurrences of terms: complex weighting algorithms are no longer needed. "Born to add" is therefore the motto of this thesis. By adding out loud, it hopefully annoys - no offence, and with all due respect - some of the well-established information retrieval approaches, like Bruce Stringbean and The Sesame Street Band annoys the Sesame Street police.

Acknowledgements

gine. Arjen de Vries did some remarkable things too. Notably, he still calls the model's probabilities "beliefs", illustrating the relativity of academic discussion, but I am mostly in his debt for implementing the language model algorithms in the Mirror DBMS, and for doing many of the experiments reported in chapter 5. Many thanks go to people who did some of the odd jobs that need to be done: Lynn Packwood did the manual disambiguation of the English queries, Thijs Westerveld implemented the interface on the corpus dictionary, Rudie Ekkelenkamp of TNO-TPD did the "high initial threshold" adaptive filtering experiments, and Dirk Heylen selflessly read the proofs of the manuscript.

This thesis would not have been written without the support of Franciska de Jong, Anton Nijholt and Wilbert Kallenberg. Both Franciska and Anton have created an open and friendly research environment, in which they leave their doors open, in which there is room for cooperation with other groups, and in which there is always support to visit distant conferences and workshops. Wilbert Kallenberg of the Faculty of Mathematical Sciences of our university played a major role in the development of the theory presented in this thesis. I would particularly like to thank him for several long brainstorm sessions, in which he convinced me to think as a 'frequencist'. I am honoured that Peter Apers, Paul Gellings, Jan Landsbergen, and Stephen Robertson agreed to complete the commission.

Special thanks go to Stephen Robertson, for making a three month internship at Microsoft Research in Cambridge possible. This thesis benefited a lot from my stay in Cambridge. I very much appreciate the help I got from Stephen Walker, Mark Hatton and David Elworthy.

Last but not least I would like to thank colleagues, family, friends, and "noabers" for forcing me to have lunch, for being patient and supportive, and just for showing interest.

Enschede, December 2000

Contents

Chapter 1

Introduction

This book introduces a new probabilistic model of information retrieval. This chapter opens with a definition of information retrieval, the introduction of the technical vocabulary used throughout the thesis, and the the theoretical and practical problems this thesis tries to solve. The chapter concludes by giving an overview of this thesis in section 1.3. Background to the research questions is given in chapters 2 and 3. Readers who want skip the introductory chapters are referred to chapter 4, which describes the new probabilistic model.

1.1 An introduction to information retrieval

Does information retrieval still need an introduction today? This book being a Ph.D. thesis, it is quite probable that you, as a reader, are already familiar with the subject. But even if you are reading this book for entertainment only, e.g. because you are a member of the author's family, chances are that you are an experienced user of information retrieval systems as well. Surveys show that about 85 % of the users of the internet use search engines to find information (Lawrence and Giles 1999). Internet search engines support the classical interactive information retrieval dialogue of entering a query, retrieving references to documents, inspecting some documents, reformulating the query, etc. People use search engines for instance to locate and buy goods, to choose a vacation destination, to select a medical treatment or to find background information on candidates of an election. A good indication of the impact of search engines and information retrieval technology on ordinary people's lifes can be found in common language. If a technology is important enough, many people will adopt the discipline's technical vocabulary and new words eventually end up in official dictionaries. For the author's native language, Dutch, information retrieval technology already left its traces in the standard language. In their latest edition, the Van Dale dictionary (Geerts and den Boon 1999) considers the Dutch word "zoekmachine" (search engine) and the originally English word "query" to be part of everyday Dutch.

Before the world wide web emerged, information storage and retrieval systems were almost exclusively used by professional indexers and searchers, e.g. for medical research, in libraries, by governmental organisations and archives. Typically, professional searchers act as 'search intermediaries' for end users or customers. They try to figure out in an interactive dialogue with the system and the customer what it is the customer needs, and how this information should be used in a successful search. Professional users differ from non-professional users because they know the collection, they know how documents in the collection are represented in the system, and they know how to use Boolean search operators to control the number of retrieved documents.[1]

Many modern information retrieval systems, like internet search engines, are specifically designed for users who are not familiar with the collection, the representation of the documents, and the use of Boolean operators. The main requirements for these systems are the following. Firstly, users should be able to enter any natural language word(s), phrase(s) or sentence(s) to the system, without the need to enter operators. This usually implies a full text information retrieval system, which is a system that potentially indexes every word in a document automatically. Secondly, the system should rank the retrieved documents by their estimated degree or probability of usefulness for the user. Thirdly, though maybe not as important as the first two, the system should support the automatic reformulation of the search statement from user feedback. These three requirements form the basis of the research presented in this thesis.

The following sections introduce the discipline of information retrieval and the technical vocabulary used throughout this thesis.

1.1.1   A definition

The discipline of information retrieval is almost as old as the computer itself. An old, if not the eldest, definition of information retrieval is the following by Mooers (1950) (recited from Savino and Sebastiani 1998).

> Information retrieval is the name of the process or method whereby a prospective user of information is able to convert his need for information into an actual list of citations to documents in storage containing information useful to him.

An information retrieval system is a software programme that stores and manages information on documents. The system assists users in finding the information they need. Unlike so-called question answering systems (Voorhees 2000), the system does not explicitly return information or answer questions. Instead, it informs on the existence and location of documents that might contain the needed information. Some suggested documents will, hopefully, satisfy the user's information need. These documents are called *relevant* documents. A perfect retrieval system would retrieve only the relevant documents and no

---

[1]Until recently, most commercial systems used the Boolean query operators AND, OR, and NOT; see section 2.2.

irrelevant document. However, perfect retrieval systems do not exist and will not exist because search statements are necessarily incomplete and relevance depends on the subjective opinion of the user. Two users may pose the same query to an information retrieval system and give different relevance judgements on the retrieved documents.

1.1.2 Basic processes of information retrieval

There are three basic processes an information retrieval system has to support: the representation of the content of the documents, the representation of the user's information need, and the comparison of the two representations. The processes are visualised in figure 1.1 (Croft 1993). In the figure, squared boxes represent data and rounded boxes represent processes.
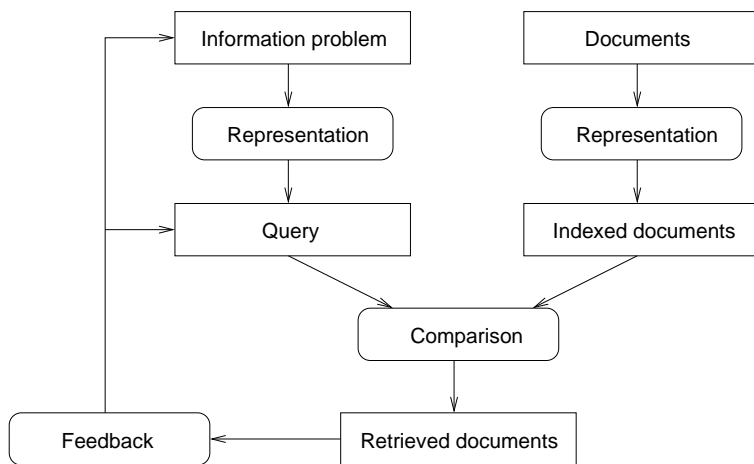
Figure 1.1: Information retrieval processes (Croft 1993)

Representing the documents is usually called the *indexing* process. The process takes place off-line, that is, the end user of the information retrieval system is not directly involved. The indexing process results in a formal representation of the document: the index representation or document representation. Often, full text retrieval systems use a rather trivial algorithm to derive the index representations, for instance an algorithm that identifies words in an English text and puts them to lower case. The indexing process may include the actual storage of the document in the system, but often documents are only stored partly, for instance only title and abstract, plus information about the actual location of the document.

The process of representing the information problem or need is often referred to as the *query formulation* process. The resulting formal representation is the query. In a broad sense, query formulation might denote the complete inter-

active dialogue between system and user, leading not only to a suitable query but possibly also to a better understanding by the user of his/her information need. In this thesis however, query formulation generally denotes the automatic formulation of the query when there are no previously retrieved documents to guide the search, that is, the formulation of the initial query. The automatic formulation of successive queries is called *relevance feedback* in this thesis. The user and the system communicate the information need by respectively queries and retrieved sets of documents. This is not the most natural form of communication. Humans would use natural language to communicate the information need amongst each other. Such a natural language statement of the information need is called a request. Automatic query formulation inputs the request and outputs an initial query. In practice, this means that some or all of the words in the request are converted to query terms, for instance by the rather trivial algorithm that puts words to lower case. Relevance feedback inputs a query or a request and some previously retrieved relevant and non-relevant documents to output a successive query. An example of the requests that were actually used in the experiments reported in this book is given in figure A.1 of the appendix.

The comparison of the query against the document representations is also called the *matching* process. The matching process results in a ranked list of relevant documents. Users will walk down this document list in search of the information they need. Ranked retrieval will hopefully put the relevant documents somewhere in the top of the ranked list, minimising the time the user has to invest on reading the documents. Simple but effective ranking algorithms use the frequency distribution of terms over documents. For instance, the words "family" and "entertainment" mentioned in the first section occur relatively infrequent in the whole book, which indicates that this book should not receive a top ranking for the request "family entertainment". Ranking algorithms based on statistical approaches easily halve the time the user has to spend on reading documents.[2] The development and evaluation of ranking algorithms is the major theme of this book.

1.2 Mathematical models of information retrieval

A mathematical model of information retrieval guides the implementation of information retrieval systems. In the traditional information retrieval systems, which are usually operated by professional searchers, only the matching process is automated; indexing and query formulation are manual processes. For these systems, mathematical models of information retrieval therefore only have to model the matching process. In practice, traditional information retrieval systems use the Boolean model of information retrieval. The Boolean model is an exact matching model, that is, it either retrieves documents or not without ranking them. The model supports the use of structured queries, which do not only contain query terms, but also relations between the terms defined by the query operators AND, OR and NOT.

---

[2]See for instance the experimental results in chapter 5.

In modern information retrieval systems, which are usually operated by non-professional users, query formulation is automated as well. However, candidate mathematical models for these systems still only model the matching process. There are many candidate models for the matching process of ranked retrieval systems. These models are so-called approximate matching models, that is, they use the frequency distribution of terms over documents to compute the ranking of the retrieved sets. Each of these models has its own advantages and disadvantages. However, there are two classical candidate models for approximate matching: the vector space model and the probabilistic model. They are classical models, not only because they were introduced already in the early 70's, but also because they represent classical problems in information retrieval. The vector space model represents the problem of ranking the documents given the initial query. The probabilistic model represents the problem of ranking the documents after some feedback is gathered.

From a practical point of view, the Boolean model, the vector space model and the probabilistic model represent three classical problems of information retrieval, respectively structured queries, initial term weighting, and relevance feedback. The Boolean model provides the query operators AND, OR and NOT to formulate structured queries. The vector space model was used by Salton and his colleagues for hundreds of term weighting experiments in order to find algorithms that predict which documents the user will find relevant given the initial query (Salton and Buckley 1988).[3] The probabilistic model, provides a theory of optimum ranking if examples of relevant documents are available.[4] The models are further described in chapter 2.

Two gaps in information retrieval theory are identified in this thesis. Firstly, none of the existing models of information retrieval account for today's top performing ranking/term weighting algorithms. Secondly, none of the existing models account for both structured queries and relevance feedback. Chapter 4 introduces a model of information retrieval that provides a well-motivated probabilistic ranking algorithm which performs as well as, or better than, today's top-performing algorithms. An extension of the model integrates structured queries and relevance feedback into one mathematical framework.

1.2.1 Automatic formulation of the initial query

The lack of mathematical models of the query formulation process is another gap in information retrieval theory that is filled by the research presented in this thesis. In practice, automatic query formulation, as used for the vector space model or the probabilistic model, often includes basic tools like stop word removal and stemming. Stop words are words in the request with little meaning, mostly function words like "the" and "it". Stemming conflates the words in the request to their stem. For instance, the stemmer introduced by Porter (1980) conflates the words "computer", "compute" and "computation" to the stem

---

[3]A term weight is a value of the term's importance in a query or a document.

[4]Examples of relevant documents, or information about the distribution of terms over relevant and nonrelevant documents, is sometimes called 'relevance information' in this book.

`comput`. An explicit model of the query formulation process should somehow give mathematical interpretations and descriptions of basic tools like stop word removal and stemming, but also any other tool that converts request words into query terms.

A model of the automatic query formulation process accounts for the fact that the vocabulary of the user might differ from the vocabulary of the indexed documents. This is sometimes called the paraphrase problem (Oard and Dorr 1996). Obviously, if the vocabulary of the indexed documents consists of stems, then the stemmer should also be used during query formulation. A less obvious example is the situation in which the user request contains synonyms of the words used in the relevant documents, for instance a request for documents about "nuclear energy", where the relevant documents all contain the words "atomic power". An extreme version of the paraphrase problem is the situation in which the language of the user request differs from the language of the documents. For instance, the user states his/her request in Dutch, but the documents are (indexed) in English. This is the problem of cross-language retrieval, which is further addressed in chapter 6. For this example, automatic query formulation results in a structured query, in which possible translations are grouped appropriately.

1.2.2   Research questions

This thesis answers the following three research questions.

1. How to apply the theory of statistical language models to three classical problems of matching models of information retrieval: initial term weighting, relevance feedback and structured queries?

2. How to apply the theory of statistical language models to the automatic formulation of structured queries from natural language search statements?

3. What can be said of the performance of the language model-based approach compared to the performance of well-established approaches?

1.3   Overview of this thesis

This book is organised as follows. Chapter 2 provides some background of the first research question. The chapter gives an overview of the most influential mathematical models that were proposed in information retrieval literature. It shows that today's top performing term weighting and ranking algorithms are not so much based on these models and theories, but instead on intuitions, approximations and on careful studies of the behaviour in test collections. Furthermore, it shows that none of the existing models accounts for both relevance feedback and structured queries.

Chapter 3 introduces the context of the second research question. The first part of the chapter describes some standard approaches to automatic query term

selection, like for instance stop word removal and stemming. The second part of the chapter describes practical query operators for the manual formulation of structured queries in modern information retrieval systems. The chapter shows that many of the query term selection strategies and query operators are not covered by any of the existing information retrieval models and theories.

Chapter 4 introduces a model of information retrieval based on the use of statistical language models. This chapter provides answers to the first and second research question above. It presents a theory of term weighting, relevance weighting using the user's feedback, and structured queries. It shows how natural language processing technology like for instance stemmers or translation modules can be interpreted and integrated for automatic query formulation. Finally, the chapter introduces a theory of advanced search facilities like proximity search and field search.

By following the methodology described in appendix A, the chapters 5, 6 and 7 provide answers to the third research question by reporting on experimental results. The basic search functionalities of the model are evaluated in chapter 5. Chapters 6 and 7 show how the new model can be applied to two themes that emerged recently: cross-language information retrieval and adaptive information filtering. The application to cross-language retrieval shows a practical use of the automatic formulation of structured queries. The application to adaptive filtering shows a practical use of the relevance feedback algorithm.

Finally, chapter 8 concludes this book by summarising the research achievements, by reflecting on the suggested approach to information retrieval, and by suggesting directions for future research.

Chapter 2

Information retrieval modelling

A short history of information retrieval modelling is given. The first section introduces the notion of mathematical models of information retrieval and explains why it is important to have these models. Section 2.2 introduces the model of exact match retrieval: the Boolean model. Section 2.3 contains a selection of the most influential models of ranked retrieval. Many of these models need an additional term weighting algorithm before they can be implemented. Section 2.4 gives a short history of term weighting.

2.1   Introduction

There are two good reasons for having models of information retrieval. The first is that it guides research and provides the means for academic discussion. The second reason is that models can serve as a blueprint to implement an actual retrieval system.

2.1.1   Models as means for discussion

Mathematical models are used in many scientific areas with the objective to understand and reason about some behaviour or phenomenon in the real world. One might for instance think of a model of our solar system that predicts the position of the planets on a particular date, or one might think of a model of the world climate that predicts the temperature given the atmospheric emissions of greenhouse gases. Webster's new collegiate dictionary (Mish et al. 1983) gives the following definition:

> **model**   a system of postulates, data and inferences presented as a
> mathematical description of an entity or state of affairs

A model of information retrieval predicts and explains what a user will find relevant given the user query. It is essential that the correctness of the model's predictions can be tested in a controlled experiment. In order to do predictions and reach a better understanding of information retrieval, models should be firmly grounded in intuitions, metaphors and some branch of mathematics. Intuitions are important because they help to get a model accepted as reasonable by the research community. Metaphors are important because they help to explain the implications of a model to a bigger audience. For instance, by comparing the earth's atmosphere with a greenhouse, non-experts will understand the implications of certain models of the atmosphere. Mathematics are essential to formalise a model, to ensure consistency, and to make sure that it can be implemented in a real system.

2.1.2   Models as a blueprint to build a system

The ability to predict user behaviour does not necessarily imply a better understanding of it. There is a more pragmatic definition of the word model that is probably more appropriate for information retrieval. A model of information retrieval might also serve as a blueprint which is used to implement an actual information retrieval system; according to Mish et al. (1983)

> **model** a pattern of something to be made

Many of the ranking algorithms and techniques presented in this chapter have the sole purpose that they should work. This goes especially for the term weighting algorithms presented in section 2.4. Of course, such algorithms and models have to fulfil certain constraints in order to be successful, that is, it has to be possible to implement them by using one of the standard information retrieval architectures for indexing and retrieval. One of those architectures is the inverted file architecture, which is currently the best choice for most applications. For completeness, the data structures and access mechanisms of the inverted file architecture will be briefly described here. Note however that mathematical models should abstract away from the implementation details presented in the next paragraph.

A naive approach to information retrieval would simply scan linearly through a collection of documents in search for the needed information. Linear scanning is appropriate when the collection is small, but for larger collections data structures are built over the text to speed up the retrieval process. An inverted file is pretty much the same thing as an index you find in the back of a book that lists index terms alphabetically together with the page numbers where they can be found. Instead of page numbers, the inverted file structure usually lists a document identifier, possibly together with the positions of the term in the document or the weight of the term in the document. Usually, the data structure is composed of two distinct substructures: the dictionary file containing the vocabulary and the postings file containing the occurrences of the terms in the collection (Harman et al. 1988; Baeza-Yates and Ribeiro-Neto 1999).

2.1.3 Three problems that models of IR have to solve

Models should be judged on the discussion aspect and the blueprint aspect. A model should be powerful in expressing complex information needs and accurate in predicting which documents will be relevant, and it should be possible to implement the model for full text information retrieval in such a way that it uses a reasonable amount of storage space and produces an answer in reasonable time, e.g. by using the inverted file architecture.

Because models serve as a vehicle for academic discussion, it is hard to give an objective account of information retrieval modelling. For instance, the use of the terms "solar system" and "greenhouse gases" in the examples above, already suggest the membership of a certain academic school. The same goes inevitably for information retrieval models. Each model uses its own specific vocabulary, which this chapter follows where possible and which is mixed with the vocabulary introduced in section 1.1 if necessary for clarification. The information retrieval models are compared to each other in a rather informal way, without an attempt to use mathematical considerations to present a taxonomy as is done by for instance Baeza-Yates and Ribeiro-Neto (1999, chapter 2), or meta-model as done by Huibers (1996). Mathematical considerations often blur the purpose of models and the problems they try to solve. Three purposes and problems are of special interest in this thesis.

1. **term weighting**, or better: models that do not simply assume the existence of a term weighting algorithm. The weight of a term is a value of the importance of a term, of which many models simply assume the existence. Term weighting is however not a trivial problem at all.

2. **relevance feedback**: Relevance feedback uses examples of relevant documents to improve the retrieval of other relevant documents.

3. **structured queries**: A structured query does not treat a query as a bag of words, but defines relations between the query terms. For some models, the support of structured queries also implies possibility to combine the evidence from different sources.

The probabilistic model presented in section 2.3.3 is an example of a model that addresses both term weighting and relevance feedback. The Bayesian networks model presented in section 2.3.8 is an example of a model that addresses structured queries and the combination of evidence. Using mathematical considerations, one might classify both models under probabilistic approaches, which is not very helpful given the difference of the models from a practical perspective.

The following sections will describe a total of eight models of information retrieval rather extensively. Many more models were suggested in information retrieval literature, but the selection made in this chapter gives a rather complete overview of the different approaches in terms of the three criteria mentioned above. Section 2.3 presents seven models of ranked retrieval. Details concerning term weighting are deferred to section 2.4. The next section presents the Boolean model. This model gets its own section because strictly speaking it is

more a model of data retrieval than a model of information retrieval, since the model does not provide a ranking of the search results. The Boolean model also deserves its own section because it served as a role model -yet another different use of the word 'model'- for many other approaches.

2.2 The Boolean model: model of models

The Boolean model is the first model of information retrieval and probably also the most criticised model. The model can be explained by thinking of a query term as a unambiguous definition of a set of documents. For instance, the query term `economic` simply defines the set of all documents that are indexed with the term `economic`. Using the operators of George Boole's mathematical logic, query terms and their corresponding sets of documents can be combined to form new sets of documents. Boole defined three basic operators, the logical product which will be called AND, the logical sum which will be called OR and the logical difference which will be called NOT (Chowdhury 1998). Combining terms with the AND operator will define a document set that is smaller than or equal to the document sets of any of the single terms. For instance, the query `social AND economic` will produce the set of documents that are indexed both with the term `social` and the term `economic`. Combining terms with the OR operator will define a document set that is bigger than or equal to the document sets of any of the single terms. So, the query `social OR political` will produce the set of documents that are indexed with either the term `social` or the term `political`, or both. This is visualised in the Venn diagrams of figure 2.1[1] in which each set


```
    social AND economic        social OR political       (social OR political)
                                                          NOT (social AND
                                                          economic)
```
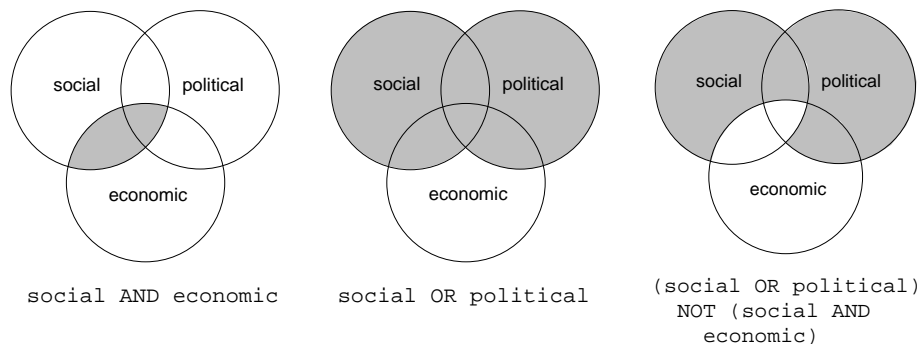
Figure 2.1: Three Boolean combinations of sets visualised as Venn diagrams

of documents is visualised by a disc. The intersections of these discs and their complements divide the document collection into 8 non-overlapping regions, the unions of which give 256 different Boolean combinations of 'social, political and economic documents'. In figure 2.1, the retrieved sets are visualised by the shaded areas.

---

[1]Often, the NOT-operator is implemented as a logical difference instead of a set complement, requiring the use of `A NOT B` instead of `A AND NOT B`

2.2.1 Advantages of the Boolean model

Although alternatives for the Boolean model have been around since the late 1960's, the Boolean model was *the* leading model for commercial retrieval systems until the mid 1990's. There are two main reasons for the predominance of Boolean retrieval. Firstly, the model gives (expert) users a sense of control over the system. It is immediately clear why a document has been retrieved given a query. If the resulting document set is either too small or too big, it is directly clear which operators will produce respectively a bigger or smaller set. Secondly, the model can be extended with proximity operators and wildcard operators in a mathematically sound way, which makes it a powerful candidate for full text retrieval systems as well. Other, more practical, reasons for the predominance of Boolean retrieval in commercial systems are the costs of major changes in software and database structures and the fact that a client community is trained on existing Boolean systems (Rasmussen 1999). The application of the Boolean retrieval model in commercial applications will be addressed further in chapter 3.

2.2.2 Disadvantages of the Boolean model

Especially for untrained users, the model has a number of clear disadvantages. Its main disadvantage is that it does not provide a ranking of retrieved documents. The model either retrieves a document or not, which might lead to the system making rather frustrating decisions. For instance, the query `social AND worker AND union` will not retrieve a document indexed with `party`, `birthday` and `cake`, but will likewise not retrieve a document indexed with `social` and `worker` that lacks the term `union`. Clearly, it is probable that the latter document is more useful than the former.

A second disadvantage is that the rigid difference between the Boolean AND and OR operators does not exist between the natural language words 'and' and 'or'. For instance, someone interested in 'social' *and* 'political' documents, should enter the query `social OR political` to retrieve all possibly interesting documents. In fact, the Boolean model is more complex than the real needs of users would justify. Expert users of Boolean retrieval systems tend to use faceted queries. A faceted query is a query that uses disjuncts of quasi-synonyms: the facets, conjoined with the AND operator. The following query for instance has two facets: `(economic OR financial OR monetary) AND (internet OR www OR portal)`. A model that defines 'phrases', like e.g. "financial portal" and 'synonyms' instead of AND and OR operators would be more natural for non-expert users of full text retrieval systems (Savino and Sebastiani 1998; Kekäläinen 1999; see also section 3.3.4).

2.2.3 Discussion

The Boolean model's main disadvantage is its inability to rank documents. For this reason, the model does not fit the needs of modern full text retrieval systems

like for instance web search engines. On the web, and for many other full text retrieval systems, ranking is of the utmost importance. Furthermore, ranking is a prerequisite of the TREC evaluation methodology used in this book. The remaining sections of this chapter discuss models that do address the need of ranking. Many of these models of ranked retrieval take some of the ideas of the Boolean model as a starting point. The Boolean model is firmly grounded in mathematics and its intuitive use of document sets provides a powerful way of reasoning about information retrieval. In this sense, the Boolean model is 'a model of models', serving as a reference point or role model for ranked retrieval models.

2.3   Models of ranked retrieval

The Boolean model's inability to rank documents is addressed by the models presented in this section. These models usually imply the use of some statistics on the terms, that is, they somehow take into account the number of occurrences of terms in the documents or in the index to compute rankings. Another key issue of models of ranked retrieval is automatic query formulation. This addresses the difficulties non-expert users have with the Boolean operators. Non-expert users should be able to enter a real natural language request, or possibly just a couple of terms, without the use of operators. Both ranking and the fact that operators are not mandatory is shared by the approaches presented in this section. For each model, some pros and cons are identified.

2.3.1   Early approaches

Luhn (1957) was the first to suggest a statistical approach to searching information. He suggested that in order to search a document collection, the inquirer should first prepare a document that is similar to the needed documents. The degree of similarity between the representation of the prepared document and the representations of the document in the collection is used to search the collection.

> The more two representations agreed in given elements and their
> distribution, the higher would be the probability of their representing
> similar information.

Following Luhn's similarity criterion, a promising first step is counting the number of elements that the query and the index representation of the document share. If the document's index representation is a vector $\vec{d} = (d_1, d_2, \cdots, d_m)$ of which each component d_k $(1 \leq k \leq m)$ is associated with an index term; and if the query is a similar vector $\vec{q} = (q_1, q_2, \cdots, q_m)$ of which the components are associated with the same terms, then the simplest of the similarity measures is

the vector inner product.[2]

$$\text{score}(\vec{d}, \vec{q}) \quad = \quad \sum_{k=1}^{m} d_k \cdot q_k \tag{2.1}$$

If the vector has binary components, i.e. the value of the component is 1 if the term occurs in the document or query and 0 if not, then the vector product measures the number of shared terms. A more general representation would use for instance natural numbers or real numbers for the components of the vectors $\vec{d}$ and $\vec{q}$.

The vector product measure does not take the size of the document and the query into account. Intuitively, longer documents will accidentally share more terms with a query, but this does not make them better candidates of relevant documents. Therefore, the vector product measure should somehow be normalised. The following similarity measures are normalised versions of the vector product measure, respectively Dice's coefficient, Jaccard's coefficient and the overlap coefficient (Van Rijsbergen 1979).

$$\text{score}(\vec{d}, \vec{q}) \quad = \quad \frac{2 \cdot \sum_{k=1}^{m} d_k \cdot q_k}{\sum_{k=1}^{m}(d_k)^2 + \sum_{k=1}^{m}(q_k)^2} \tag{2.2}$$

$$\text{score}(\vec{d}, \vec{q}) \quad = \quad \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sum_{k=1}^{m}(d_k)^2 + \sum_{k=1}^{m}(q_k)^2 - \sum_{k=1}^{m} d_k \cdot q_k} \tag{2.3}$$

$$\text{score}(\vec{d}, \vec{q}) \quad = \quad \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\min(\sum_{k=1}^{m}(d_k)^2, \; \sum_{k=1}^{m}(q_k)^2)} \tag{2.4}$$

All these measures are clearly based on ad-hoc considerations. They lack a strong metaphor of searching and strong support from some branch of mathematics. They are commonly regarded to be obsolete (Savino and Sebastiani 1998).

2.3.2   The vector space model

Salton and McGill (1983) suggested a model based on Luhn's similarity criterion that has a stronger theoretical motivation. They considered the index representations and the query as vectors embedded in a high dimensional Euclidean space, where each term is assigned a separate dimension. The similarity measure is usually the cosine of the angle that separates the two vectors $\vec{d}$ and $\vec{q}$. The cosine of an angle is 0 if the vectors are orthogonal in the multidimensional space and 1 if the angle is 0 degrees.

$$\text{score}(\vec{d}, \vec{q}) \quad = \quad \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m}(d_k)^2} \cdot \sqrt{\sum_{k=1}^{m}(q_k)^2}} \tag{2.5}$$

The metaphor of angles between vectors in a multidimensional space makes it easy to explain the implications of the model to non-experts. Up to three

---

[2]The function that computes a value that is used to rank documents, and the value itself, will be called the (document) score.

dimensions, one can easily visualise the document and query vectors. Figure 2.2 visualises an example document vector and an example query vector in the space that is spanned by the three terms `social`, `economic` and `political`. The intuitive geometric interpretation makes it relatively easy to apply the model to new information retrieval problems. The vector space model guided research in for instance automatic text categorisation and document clustering.
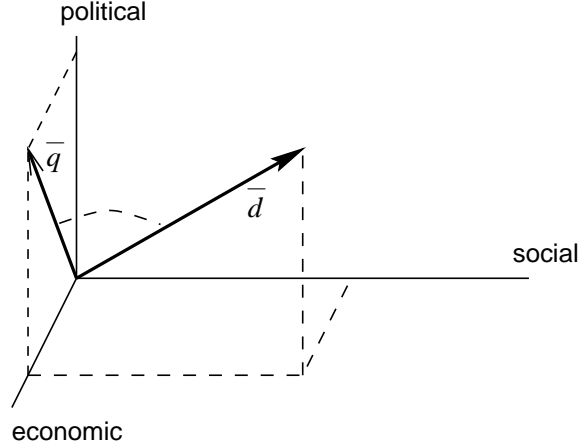
Figure 2.2: A query and document representation in the vector space model

Normalisation of vector lengths

Measuring the cosine of the angle between vectors is equivalent with normalising the vectors to unit length and taking the vector inner product. If index representations and queries are properly normalised, then the vector product measure of equation 2.1 does have a strong theoretical motivation.

$$\text{score}(\vec{d}, \vec{q}) \;=\; \sum_{k=1}^{m} n(d_k) \cdot n(q_k) \quad \text{where } n(v_k) = \frac{v_k}{\sqrt{\sum_{k=1}^{m}(v_k)^2}} \tag{2.6}$$

Some rather ad-hoc, but quite successful retrieval algorithms are nicely grounded in the vector space model if the vector lengths are normalised. An example is the relevance feedback algorithm by Rocchio (1971). Rocchio suggested the following algorithm for relevance feedback, where $\vec{q}_{old}$ is the original query, $\vec{q}_{new}$ is the revised query, $\vec{d}_{rel}^{(i)}$ ($1 \leq i \leq r$) is one of the r documents the user selected as relevant, and $\vec{d}_{nonrel}^{(i)}$ ($1 \leq i \leq n$) is one of the n documents the user selected as non-relevant.

$$\vec{q}_{new} \;=\; \vec{q}_{old} \;+\; \frac{1}{r}\sum_{i=1}^{r} \vec{d}_{rel}^{(i)} \;-\; \frac{1}{n}\sum_{i=1}^{n} \vec{d}_{nonrel}^{(i)} \tag{2.7}$$

The normalised vectors of documents and queries can be viewed at as points on a hypersphere at unit length from the origin. In equation 2.7, the first sum calculates the centroid of the points of the known relevant documents on the hypersphere. In the centroid, the angle with the known relevant documents is minimised. The second sum calculates the centroid of the points of the known non-relevant documents. Moving the query towards the centroid of the known relevant documents and away from the centroid of the known non-relevant documents is guaranteed to improve retrieval performance. The sphere is visualised for two dimensions in figure 2.3. The figure is taken from Savino and Sebastiani (1998).
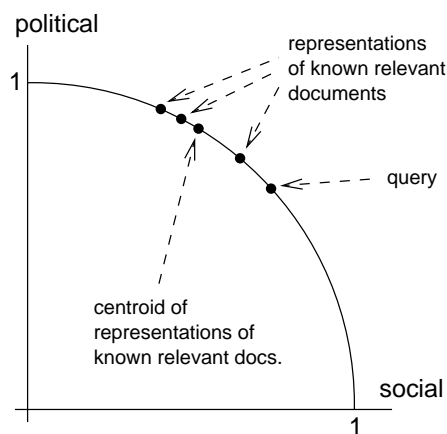
Figure 2.3: Rocchio's relevance feedback method

Discussion

The main disadvantage of the vector space model is that it does not in any way subscribe what the values of the vector components should be. Early experiments by Salton (1971) already suggested that term weighting is not a trivial problem at all. Term weighting approaches are addressed in section 2.4. A second disadvantage of the vector space model is that it is not possible to include term dependencies into the model, for instance for modelling of phrases or adjacent terms. It is however possible to give a geometric interpretation of Boolean-structured queries, which is described in section 2.3.5. A third problem with the vector space model is its implementation. The calculation of the cosine measure needs the values of all vector components, but these are not available in a inverted file architecture. In practice, the normalised values and the vector product algorithm have to be used. Either the normalised weights have to be stored in the inverted file, or the normalisation values have to be stored separately. Both take significantly more storage space than would be required for the Boolean model (Witten, Moffat, and Bell 1994).

2.3.3 The probabilistic model

Maron and Kuhns (1960) formulated a criterion that implicitly goes against Luhn's idea to use the degree of similarity between index representations and query. They argued that a retrieval system should rank the documents in the collection in order of their probability of relevance. Robertson (1977) called the criterion the 'probability ranking principle'. He formulated the principle, which he contributed to William Cooper, as follows.

> If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data.

This seems a rather trivial requirement indeed, since the objective of information retrieval systems is defined in chapter 1 as "to help the user to find relevant documents". However, Robertson showed that optimality of ranking by the probability of relevance can only be guaranteed if the following conditions are met. Firstly, relevance should be a dichotomous variable, either yes or no. Secondly, relevance of a document to a request should not depend on the other documents in the collection.

The probability of relevance

Whereas Luhn's intuitive similarity criterion raises the question: "What exactly makes two representations similar?", Robertson's probability ranking principle raises the question: "How, and on the basis of what data, should the probability of relevance be estimated?". First it is necessary to make the notion of 'probability of relevance' explicit. Robertson adopted the Boolean model's viewpoint by looking at a term as a definition of a set of documents. Suppose a user enters a query containing a single term, for instance the term `social`. If all documents that fulfil the user's need were known, it would be possible to divide the document collection into 4 non-overlapping subsets as visualised in the Venn diagram of figure 2.4. The figure contains additional information about the size of each of the non-overlapping subsets. The collection in question has 10,000 documents, of which 1,000 contain the word "social"; only 11 documents are relevant to the query of which 1 contains the word "social". If a document is taken at random from the set of documents that are indexed with `social`, then the probability of picking a relevant document is $1 / 1,000 = 0.0010$. If a document is taken at random from the set of documents that are *not* indexed with `social`, then the probability of relevance is bigger: $10 / 9,000 = 0.0011$. Since the user entered only one index term, the system has only two options: either the documents indexed with the term are presented first in the ranking, or the documents that are not indexed with the term are presented first. In

the example of figure 2.4, it is wise to present the user first with documents that not are indexed with the query term `social`, that is, to present first the documents that are 'dissimilar' to the query. Clearly, such a strategy violates Luhn's similarity criterion.
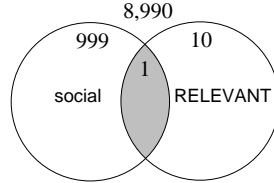
Figure 2.4: Venn diagram of the collection given the query term `social`

The binary independence assumption

If the user enters two terms, for instance the terms `social` and `political`, then there are four sets that must find their place in the final ranking: `social AND political`, `social NOT political`, `political NOT social` and `NOT(social OR political)`. Each of these Boolean subsets can be represented by a pair of binary values, the first value indicating whether the subset includes documents indexed with `social`, the second value indicating whether the subset includes documents indexed with `political`. The four Boolean subsets are represented by respectively $(1, 1)$, $(1, 0)$, $(0, 1)$ and $(0, 0)$. If the documents that fulfil the user's need were known, it would be possible to calculate the probability of picking a relevant document from each of these subsets and rank the subsets accordingly.

Unfortunately, the number of non-overlapping subsets increases exponentially with the number of query terms. To make the computation of the probability of relevance ranking possible in reasonable time, the probability of relevance in the complex subsets is determined from the probability of relevance in the sets of the single terms. It is assumed that given relevance (or non-relevance) terms occur independently from each other in the documents. The independence assumption can be applied as follows. Let L be the random variable "document is relevant" with a binary sample space $\{0, 1\}$, 1 indicating relevance and 0 non-relevance. Let D_k $(1 \leq k \leq n)$ be a random variable indicating "document belongs to the subset indexed with the kth query term" with a binary sample space $\{0, 1\}$. Given a query of length n, documents in every subset $D_1, D_2, \cdots, D_n$ will be assigned the value defined by equation 2.9, and the subsets should be ranked accordingly (Robertson and Sparck-Jones 1976; Van Rijsbergen 1979). Note that duplicate query terms retrieve the same subset of documents and should be ignored in the formulas below.

$$\text{logit } P(L\!=\!1|D_1,\cdots,D_n) \quad = \quad \log \frac{P(L=1|D_1,\cdots,D_n)}{P(L=0|D_1,\cdots,D_n)} \tag{2.8}$$

$$\begin{aligned}
&= \; \log \frac{P(L=1)P(D_1,\cdots,D_n|L=1)\,/\,P(D_1,\cdots,D_n)}{P(L=0)P(D_1,\cdots,D_n|L=0)\,/\,P(D_1,\cdots,D_n)} \\[2mm]
&= \; \log \frac{P(D_1,\cdots,D_n|L=1)}{P(D_1,\cdots,D_n|L=0)} + \operatorname{logit} P(L=1) \\[2mm]
&= \; \sum_{k=1}^{n} \log \frac{P(D_k|L=1)}{P(D_k|L=0)} + \operatorname{logit} P(L=1)
\end{aligned} \tag{2.9}$$

Equation 2.8 is a variation of Bayes' rule that uses a logistic transformation of probabilities, which is defined by $\operatorname{logit} P(L) = \log(P(L)\,/\,(1 - P(L)))$. It is used to put the equation in a convenient linear form. The transformation is strictly monotonic, so ranking documents by equation 2.8 will in fact rank them by the probability of relevance. The conditional independence assumption is formalised in equation 2.9. Often, equation 2.9 is called the binary independence assumption, because both $P(D_k|L=1)$ and $P(D_k|L=0)$ are explicitly present in the formula. Because of the independence assumption, the definition of probabilities as proportions as shown in figure 2.4 is no longer possible if more than one query term is present. So, the probability of relevance of the subset of documents that are indexed with both `social` and `political` is not necessarily the number of relevant documents in this subset divided by the size of the subset.

Implementation

Equation 2.9 needs some computation for subsets for which $D_k = 0$, that is for non-matching query terms. In the vector space model non-matching terms are assigned zero weight, which is usual convenient for implementation reasons. Therefore, $\sum_{k=1}^{n} \log(P(D_k = 0|L = 1)\,/\,P(D_k = 0|L = 0))$ is subtracted from the score of each document subset. This does not affect the ranking of the documents and assigns a score of zero to documents with no matching terms.

$$P(L=1|D_1,\cdots,D_n) \;\; \propto \; \sum_{\substack{k \,\in\, \text{match-} \\ \text{ing terms}}} \log \frac{P(D_k=1|L=1)\,P(D_k=0|L=0)}{P(D_k=1|L=0)\,P(D_k=0|L=1)} \tag{2.10}$$

The probabilities are defined by the relative sizes of the of the subsets of documents that are indexed by the query terms. Figure 2.5 shows again the Venn diagram of documents indexed with `social`. The size of the non-overlapping subsets are defined by R: the number of relevant documents, n_k: the number of documents indexed with `social`, r_k: the number of relevant documents that are indexed with `social` and N: the total number of documents in the collection. The values of n_k and N are available to the system, but the values of r_k and R are only available if the user provides those to the system, typically by marking some previously retrieved documents as relevant. If r_k and R are not available to the system, it is necessary to make some assumptions about them. Robertson and Sparck-Jones (1976) simply add 0.5 to each non-overlapping subset and Croft and Harper (1979) assume a constant value for $P(D_k|L=1)$. If

$$P(D_k{=}1\,|\,L{=}1) \ = \ {}^{r_k}\!/_R$$

$$P(D_k{=}1\,|\,L{=}0) \ = \ {}^{n_k-r_k}\!/_{N-R}$$

$$P(D_k{=}0\,|\,L{=}1) \ = \ {}^{R-r_k}\!/_R$$

$$P(D_k{=}0\,|\,L{=}0) \ = \ {}^{N-n_k-R+r_k}\!/_{N-R}$$
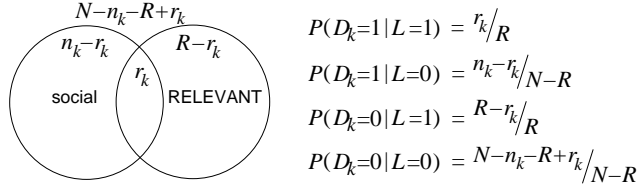
Figure 2.5: Definition of probabilities

the additional assumption is made that the number of relevant documents is much smaller than the size of the collection, more specifically: $R, r_k \ll N, n_k$, then documents might be ranked by a *idf*-like measure: $\log((N-n_k)\,/\,n_k)$ (see section 2.4).

Discussion

The probabilistic model is one of the few retrieval models that does not need an additional term weighting algorithm to be implemented (see section 2.4). Ranking algorithms are completely derived from theory. The probabilistic model has been one of the most influential retrieval models for this very reason. Unfortunately, in many applications the distribution of terms over relevant and non-relevant documents will not be available. In these situations probability of relevance estimation is of theoretical interest only.

The main disadvantage of the probabilistic model is that it only defines a partial ranking of the documents. For short queries, the number of different subsets will be relatively low. By looking at a term as a definition of a set of documents, the probabilistic model ignores the distribution of terms within documents. In fact, one might argue that the probabilistic model suffers partially from the same defect as the Boolean model. It does not allow the user to really control the retrieved set of documents. For short queries it will not seldomly assign the same rank to, for instance, the first 100 documents retrieved.

Many more probabilistic approaches have been suggested (Fuhr 1992). Two of those models will be discussed in section 2.3.6 and 2.3.8.

2.3.4 Fuzzy set models

In fuzzy set theory (Zadeh 1965) an element has a degree of membership to a set. Whereas in the Boolean model documents belong either to the set defined by an index term or not, in the fuzzy set model documents belong with a given degree to the set defined by an index term. The degree of membership is used to represent inexactness or vagueness. The idea is the following. Although it is known with certainty that a document contains a term, for instance the term `economic`, some documents are more economic than others. For the degree of membership T of a single term, one of the document term weighting formulas of section 2.4 can be used. The rules for the membership function T of the union

and intersection of fuzzy sets are usually the following.

$$
\begin{aligned}
T(\texttt{a AND b}) &= \min(T(\texttt{a}), T(\texttt{b})) \\
T(\texttt{a OR b}) &= \max(T(\texttt{a}), T(\texttt{b})) \\
T(\texttt{NOT b}) &= 1 - T(\texttt{b})
\end{aligned}
\tag{2.11}
$$

These operators are not very effective for the following reason. Suppose a query a OR b is entered, then a document belonging to the fuzzy set of a with $T(\texttt{a}) = 0.8$ and to b with $T(\texttt{a}) = 0.7$ will get the same score as a document belonging to a with $T(\texttt{a}) = 0.8$ and to b with $T(\texttt{a}) = 0.1$. Intuitively, one would rank the first document above the second in the example. A similar example can be constructed for the intersection of fuzzy sets.

The operators of equation 2.11 are not the only generalisations of the strict Boolean set operators. A variety of fuzzy set operators have been developed since the late 1970's (Lee 1995). An example of an extension of the Boolean model that is at least inspired by fuzzy set theory, is the model of Paice (1984). Paice's set operator take into account all of the document weights in the final score, not only the maximum or minimum weight. The score of a document given a query (a_1 AND a_2 AND $\cdots$ AND a_n) or a query (a_1 OR a_2 OR $\cdots$ OR a_n) is computed as follows:

$$
\text{score} = \frac{\sum_{k=1}^{n} r^{i-1} T(a_k)}{\sum_{k=1}^{n} r^{i-1}}
\tag{2.12}
$$

where the $T(a_k)$'s are considered in descending order for OR queries and in ascending order for AND queries. For Boolean queries with more than one operator, the evaluation proceeds recursively from the innermost clause. The value of r has to be determined experimentally for both set operators. It determines the 'softness' of the operator. For values close to one, the operators show similar behaviour. For large values, the operators behave more like in the Boolean model.

Fuzzy set models have the advantage over the vector space model and the probabilistic model that they provide a ranking for structured queries. An extensive comparison both in terms of theoretical properties and retrieval effectiveness of fuzzy set models and other extended Boolean models was conducted by Lee (1995). Lee measured retrieval effectiveness of Boolean queries on one of the TREC subcollections. The best performing extended Boolean models were Paice's model and the p-norm model which will be addressed in the next section. Like the vector space model, the fuzzy set models need an additional term weighting algorithm to determine the membership function of single terms. A related disadvantage of the fuzzy set models is that they do not give insight in why some operators perform better than others. It is not clear what the intuitions behind the models are. The theory gives little guidance in how to apply a fuzzy set model to new retrieval problems.

2.3.5 The p-norm extended Boolean model

The p-norm extended Boolean model was developed by Salton, Fox, and Wu (1983), following the vector space model's metaphor of documents in a multi-dimensional Euclidean space. If the two terms `social` and `political` are again considered, the vector space spanned by the terms can be easily visualised. If document vectors are normalised to unit length, then the point (1,1) in the space represents the situation that both terms are present with weight 1. This is the desirable location for a document matching the query `social AND political`. For the query `social OR political` on the other hand, the point (0,0) representing the situation that both terms are absent, is the undesirable location for a document. Therefore, AND-queries should rank documents in order of increasing distance from the point (1,1) and OR-queries in order of decreasing distance from the point (0,0). If the distances are properly normalised to fall between 0 and 1, then the following formulas apply. In the formula d_a denotes the weight of the term a in a document with index representation $\vec{d}$.

$$
\begin{aligned}
\text{score}(\vec{d}, a \ \text{OR} \ b) &= \sqrt{\frac{(d_a - 0)^2 + (d_b - 0)^2}{2}} \\
\text{score}(\vec{d}, a \ \text{AND} \ b) &= 1 - \sqrt{\frac{(1 - d_a)^2 + (1 - d_b)^2}{2}}
\end{aligned}
\tag{2.13}
$$

Salton, Fox, and Wu (1983) suggested two generalisations of the basic idea. First of all, query term weights were included to reflect the importance of individual terms. Secondly, the Euclidean distance measures were generalised by introducing a parameter p for each set operator. The resulting p-norm model uses the following formulas.

$$
\begin{aligned}
\text{score}(\vec{d}, \vec{q}_{\text{OR}(p)}) &= \left(\frac{\sum_{k=1}^{m} (q_k)^p (d_k)^p}{\sum_{k=1}^{m} (q_k)^p} \right)^{1/p} \\
\text{score}(\vec{d}, \vec{q}_{\text{AND}(p)}) &= 1 - \left(\frac{\sum_{k=1}^{m} (q_k)^p (1 - d_k)^p}{\sum_{k=1}^{m} (q_k)^p} \right)^{1/p}
\end{aligned}
\tag{2.14}
$$

The introduction of p results in similar softness of Boolean operators as in Paice's formula. For $p = 2$ the formulas will use the Euclidean distance measures as in equation 2.13. For $p = 1$ the OR-operator and the AND-operator produce the exact same results and the model behaves like the vector space model. If $p \to \infty$ then the ranking is evaluated according to the standard fuzzy set operators of equation 2.11.

As said in section 2.3.4, the p-norm model belongs to the best performing extended Boolean models. Based on recent publications about such models, the p-norm model is probably more popular for extended Boolean retrieval than other well-performing algorithms. Greiff, Croft, and Turtle (1997) copied the behaviour of the p-norm model in their inference network architecture and Losada and Barreiro (1999) propose a belief revision operator that is equivalent to a p-norm case (see also section 2.3.8).

A disadvantage of the p-norm model is that it needs an additional term weighting algorithm to be implemented.

2.3.6   The 2-Poisson model

Bookstein and Swanson (1974) studied the problem of developing a set of statistical rules for the purpose of identifying the index terms of a document. They suggested that the number of occurrences tf of terms in documents could be modelled by a mixture of two Poisson distributions as follows, where X is a random variable for the number of occurrences.

$$P(X = tf) \;=\; \lambda \frac{e^{-\mu_1}\,(\mu_1)^{tf}}{tf!} + (1-\lambda)\frac{e^{-\mu_2}\,(\mu_2)^{tf}}{tf!} \qquad (2.15)$$

The model assumes that the documents were created by a random stream of term occurrences. For each term, the collection can be divided into two subsets. Documents in subset one treat a subject referred to by a term to a greater extent than documents in subset two. This is represented by λ which is the proportion of the documents that belong to subset one and by the Poisson means μ_1 and μ_2 ($\mu_1 \geq \mu_2$) which can be estimated from the mean number of occurrences of the term in the respective subsets. For each term, the model needs these three parameters, but unfortunately, it is unknown to which subset each document belongs. The estimation of the three parameters should therefore be done iteratively by applying e.g. the expectation maximisation algorithm (see also section 4.4) or alternatively by the method of moments as done by Harter (1975).

If a document is taken at random from subset one, then the probability of relevance of this document is assumed to be equal to, or higher than, the probability of relevance of a document from subset two; because the probability of relevance is assumed to be correlated with the extent to which a subject referred to by a term is treated, and because $\mu_1 \geq \mu_2$. Useful terms will make a good distinction between relevant and non-relevant documents, that is, both subsets will have very different Poisson means μ_1 and μ_2. Therefore, Harter (1975) suggests the following measure of effectiveness of an index term that can be used to rank the documents given a query.

$$z \;=\; \frac{\mu_1 - \mu_2}{\sqrt{\mu_1 + \mu_2}} \qquad (2.16)$$

The 2-Poisson model's main advantage is that it does not need an additional term weighting algorithm to be implemented. In this respect, the model contributed to the understanding of information retrieval and inspired some researchers in developing new models as shown in the next section. The model's biggest problem, however, is the estimation of the parameters. For each term there are three unknown parameters that cannot be estimated directly from the observed data. Furthermore, despite the model's complexity, it still might not fit the actual data if the term frequencies differ very much per document. Some studies therefore examine the use of more than two Poisson functions, but this makes the estimation problem even more intractable (Margulis 1993).

2.3.7 An extension of the probabilistic model

Robertson, Van Rijsbergen, and Porter (1981) used the 2-Poisson model to include the number of term occurrences in the probabilistic model. First of all, they redefined D_k as a random variable which has as its sample space the set of natural numbers $\{0, 1, 2, \cdots\}$, indicating "document belongs to the subset of documents with d_k occurrences of the kth query term". Following similar considerations as in section 2.3.3, this results in the following weighting algorithm that only uses the matching terms in its computation (so, $d_k > 0$).

$$P(L=1|D_1, \cdots, D_n) \;\; \propto \sum_{\substack{k \,\in\, \text{match-}\\ \text{ing terms}}} \log \frac{P(D_k=d_k|L=1)\,P(D_k=0|L=0)}{P(D_k=d_k|L=0)\,P(D_k=0|L=1)} \quad (2.17)$$

Subsequently, (Robertson et al. 1981) assumed that the number of term occurrences d_k in the relevant and non-relevant documents can be modelled by the 2-Poisson distribution. For each term, documents that belong to subset one are called "elite" for that term. For effective index terms, the proportion of the relevant documents that are elite for that term should differ from the proportion of the non-relevant document that are elite for that term. In the following formulas, λ_k is taken as the probability of eliteness given relevance and κ_k is taken as the probability of eliteness given non-relevance.

$$\begin{aligned} P(D_k=d_k|L=1) &= \lambda_k \frac{e^{-\mu_{1k}}\,(\mu_{1k})^{d_k}}{d_k!} + (1-\lambda_k)\frac{e^{-\mu_{2k}}\,(\mu_{2k})^{d_k}}{d_k!}\\[2mm] P(D_k=d_k|L=0) &= \kappa_k \frac{e^{-\mu_{1k}}\,(\mu_{1k})^{d_k}}{d_k!} + (1-\kappa_k)\frac{e^{-\mu_{2k}}\,(\mu_{2k})^{d_k}}{d_k!} \end{aligned} \quad (2.18)$$

So, relevance is related to eliteness rather than directly to the frequency d_k of the kth query term. The frequency in turn is assumed to depend only on eliteness, but not on relevance. The probabilistic model's assumption that terms occur independently in documents given (non-)relevance should be augmented with the assumption that the eliteness properties of the terms are independent as well.

The model's main advantage is again that it does not need an additional term weighting algorithm. However, whereas Harter (1975) had to estimate three parameters for each term, the extension of the probabilistic model needs the estimation of four parameters for each term, for none of which there will be any direct evidence. Furthermore, by using the probability of relevance in the subset of documents with d_k occurrences of the query term, the model implicitly assumes that all documents have equal lengths, which is rarely the case.[3] Although actual implementation of the extension of the probabilistic model is cumbersome, practical weighting algorithms have been suggested that are rough approximations of the model (see section 2.4.6).

---

[3]In fact, Harter (1975) also assumed equal document lengths for his application of the 2-Poisson model.

2.3.8   Bayesian network models

A Bayesian network is an acyclic directed graph that encodes probabilistic dependency relationships between random variables. A directed graph is acyclic if there is no directed path $A \to \cdots \to Z$ such that $A = Z$. The presentation of probability distributions as directed graphs, makes it possible to analyse complex conditional independence assumptions by following a graph theoretic approach. Probability theory ensures that the system as a whole is consistent. Some alternative names for Bayesian networks are belief networks, probabilistic independence networks, influence diagrams and causal nets (Pearl 1988). This is further explained by the following simple model suggested by Turtle (1991) and Turtle and Croft (1992). A similar approach is suggested by Ribeiro and Muntz (1996).
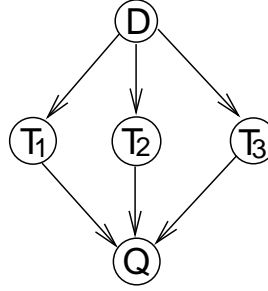
Figure 2.6: Simple Bayesian network

The Bayesian network of figure 2.6 shows Turtle's simple model of the relevance of a document given a query of three non-equal terms, for instance the example query `social political economic`. All nodes in the network represent binary random variables with values $\{0, 1\}$. The event "query is fulfilled" ($Q = 1$) has three possible causes: the subject referred to by the term `social` is true ($T_1 = 1$), or the subject referred to by the term `political` is true ($T_2 = 1$), or the subject of `economic` is true ($T_3 = 1$), or a combination of the three causes. The three subjects in turn are inferred from the event "document is relevant" ($D = 1$). By the chain rule of probability, the joint probability of all the nodes in the graph above is:

$$P(D, T_1, T_2, T_3, Q) = \atop P(D)P(T_1|D)P(T_2|D, T_1)P(T_3|D, T_1, T_2)P(Q|D, T_1, T_2, T_3) \quad (2.19)$$

The directions of the arcs suggest the dependence relations between the random variables. The model makes the following conditional independence assumptions.

$$P(D, T_1, T_2, T_3, Q) = P(D)\, P(T_1|D)\, P(T_2|D)\, P(T_3|D)\, P(Q|T_1, T_2, T_3) \quad (2.20)$$

The second, third and fourth term in equation 2.19 are simplified because T_1, T_2 and T_3 are independent given their parent D. The last term is simplified because Q is independent of D given its parents T_1, T_2 and T_3. Now, the network should be used as follows. If it is hypothesised that the document is relevant ($D = 1$), the probability of query fulfilment $P(Q = 1|D = 1)$ can be used as a score to rank the documents. The joint probability distribution defined by equation 2.19 can be used as follows to calculate the score.

$$P(Q{=}1|D{=}1) = \frac{P(Q{=}1, D{=}1)}{P(D{=}1)} =$$
$$\frac{\sum_{t_1,t_2,t_3} P(D{=}1, T_1{=}t_1, T_2{=}t_2, T_3{=}t_3, Q{=}1)}{P(D{=}1)}$$

The only thing that's still missing is the specification of the probabilities. Possible strengths of the relationships are shown in the following five tables.

			$P(D = 0)$	$P(D = 1)$		D	$P(T_1 = 0)$	$P(T_1 = 1)$
			0.999	0.001		0	0.60	0.40
						1	0.05	0.95
T_1	T_2	T_3	$P(Q = 0)$	$P(Q = 1)$				
0	0	0	1.000	0.000		D	$P(T_2 = 0)$	$P(T_2 = 1)$
0	0	1	0.901	0.099		0	0.88	0.12
0	1	0	0.887	0.113		1	1.00	0.00
0	1	1	0.992	0.008				
1	0	0	0.547	0.453		D	$P(T_3 = 0)$	$P(T_3 = 1)$
1	1	0	0.332	0.664		0	0.97	0.03
1	0	1	0.271	0.729		1	0.02	0.98
1	1	1	0.220	0.780				

Figure 2.7: Example specification of the model's parameters

The table of $P(Q|T_1, T_2, T_3)$ shows a potential difficulty of this network. The number of probabilities that have to be specified for a node grows exponentially with its number of parents, so a query of n non-equal terms requires the specification of 2^{n+1} possible values of $P(Q|T_1, T_2, \cdots, T_n)$. Despite the simplifying assumptions made by the conditional independencies, the model has to make additional simplifying assumptions to make it possible to calculate the probability in reasonable time. Turtle (1991, page 53) therefore suggests the use of four canonical forms of $P(Q|T_1, T_2, \cdots, T_n)$ which can be computed on the fly in linear time. The four canonical forms which are called "and", "or", "sum" and "weighted sum" ("wsum" for short), are displayed in figure 2.8. The weights w_1, w_2 and w_3 in the last columns are restricted to positive values and should sum up to one.[4] Suppose for now that the values of $P(T_1|D)$, $P(T_2|D)$ and $P(T_3|D)$ are known and given by p_1, p_2 and p_3. The calculation of $P(D = 1|Q = 1)$ by

---

[4]The definition of "wsum" in (Turtle 1991) is more general

T_1	T_2	T_3	$P_{\mathrm{and}}(Q)$ 0	1	$P_{\mathrm{or}}(Q)$ 0	1	$P_{\mathrm{sum}}(Q)$ 0	1	$P_{\mathrm{wsum}}(Q)$ 0	1
0	0	0	1	0	1	0	1	0	1	0
0	0	1	1	0	0	1	2/3	1/3	$1-w_3$	w_3
0	1	0	1	0	0	1	2/3	1/3	$1-w_2$	w_2
0	1	1	1	0	0	1	1/3	2/3	$1-w_2-w_3$	w_2+w_3
1	0	0	1	0	0	1	2/3	1/3	$1-w_1$	w_1
1	0	1	1	0	0	1	1/3	2/3	$1-w_1-w_3$	w_1+w_3
1	1	0	1	0	0	1	1/3	2/3	$1-w_1-w_2$	w_1+w_2
1	1	1	0	1	0	1	0	1	0	1

Figure 2.8: Canonical forms of $P(Q|T_1, T_2, T_3)$

the canonical forms of table 2.8 will give the same results as the to the following calculations, which only require linear time. Note that the weighted sum equals the vector product algorithm of equation 2.1.

$$
\begin{aligned}
P_{\mathrm{and}}(Q = 1|D = 1) &= p_1\, p_2\, p_3 \\
P_{\mathrm{or}}(Q = 1|D = 1) &= 1 - (1-p_1)(1-p_2)(1-p_3) \\
P_{\mathrm{sum}}(Q = 1|D = 1) &= (p_1 + p_2 + p_3)\, /\, 3 \\
P_{\mathrm{wsum}}(Q = 1|D = 1) &= w_1\, p_1 + w_2\, p_2 + w_3\, p_3
\end{aligned}
\tag{2.21}
$$

The main advantage of the Bayesian network models suggested by Turtle and Croft (1992) is that the the network topology can be used to combine evidence in a complex way. Many other recent approaches to information retrieval seek for new ways of combining evidence from multiple sources (e.g. Van Rijsbergen 1986; Sebastiani 1994; Fuhr 1995; Wong and Yao 1995) Figure 2.9 shows such a complex Bayesian network. In the network R_1 and R_2 define different representations of the document, for instance one might represent the document's title words, whereas the other might represent words from the abstract. The model's probabilities might indicate that title words are more important than words from the abstract. The nodes Q_1, Q_2 and Q_3 represent different queries for the same information need, which is represented by the node I. The query represented by Q_2 is evaluated as `or(and(T_1 T_2) T_3))`, whereas the query Q_3 is evaluated as `wsum(T_1 T_2 T_3)`.

There are two disadvantages of the Bayesian network models presented in this section. Firstly, the approaches do not suggest how the probability measures $P(T_i|D)$, $(1 \leq i \leq n)$ should be estimated. Instead, the approaches suggest the use of Bayesian probabilities. In a nutshell, the Bayesian probability of an event is a person's degree of belief in that event, which does not have to refer to a physical mechanism or experiment. In contrast, the classical probability always implies such an experiment and therefore can always be interpreted as a relative frequency. Considering probabilities as a person's degree of belief is quite practical if a medical expert system is built as e.g. described by Heckerman (1991). For full text information retrieval systems however, experts are by
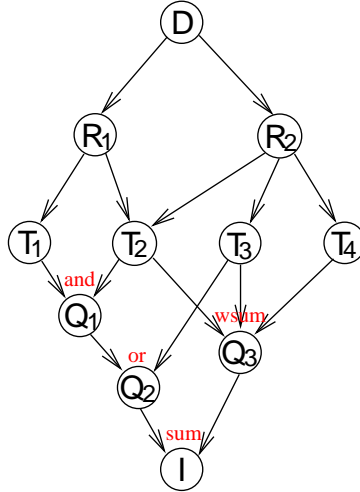
Figure 2.9: Complex Bayesian network

definition not available for specifying the probabilities of the network because it implies manual indexing of the collection. The models therefore use one of the term weighting algorithms that use term frequencies and document frequencies as presented in section 2.4. Note that, despite of the name, the use of Bayesian networks is not restricted to Bayesian probabilities (Jordan 1998; Krause 1998).

A second disadvantage of the Bayesian network models presented in this section is that the calculation of the probabilities generally takes exponential time in the number n of non-equal query terms. The introduction of the four canonical forms solves this problem, but it could have been solved by the network topology. For instance the definition of P_{and} in equation 2.21 actually suggests (conditional) independence between the probabilities p_1, p_2 and p_3 and, for instance the definition of P_{wsum} suggests the use of a mixture model topology (Jordan 1998). By using the four canonical forms, the network is tractable if it is used for inference, but it is still intractable if used for updating the probabilities. Updating the probabilities might be an effective approach to relevance feedback. Although the Bayesian network formalism comes with efficient learning algorithms, these algorithms can in practice not be applied in reasonable time on the network model presented in this section (Turtle and Croft 1991).

2.4 Term weighting experiments

Of the models presented in section 2.3, the vector space model, the fuzzy set models, the p-norm model and the Bayesian network models all need an additional term weighting algorithm before they can be implemented. Weighting of

search terms is the single most important factor in the performance of information retrieval systems. The development of term weighting approaches is as much an art as it is a science: Literally thousands of term weighting algorithms were used experimentally during the last 25 years, especially within the Smart projects. Although Salton experimented with term weighting in the 1960's, this section starts its history of term weighting in the 1970's covering important work until the end of the 1990's. The early history presented in this section is largely based on similar overviews of Belkin and Croft (1987) and Harman (1992). The section uses the notation that was introduced for the vector space model in section 2.3.2. Unless stated otherwise, weights of terms that do not occur in the document or the query are zero.

2.4.1   *idf* weighting

The document frequency *df* of a term is defined by the number of documents a term occurs in. A term with a low document frequency is more specific than a term with a high document frequency. Sparck-Jones (1972) suggested that therefore, the system should treat matches on non-frequent terms as more valuable than ones on frequent terms. An intuitive way to relate the matching value of a term to its document frequency is suggested by a Zipf-like distribution of words in a vocabulary (see e.g. Manning and Schütze 1999). If $f(df) = m$ such that $2^{m-1} < df \leq 2^m$, and N is the number of documents in the collection, then the weight of a term that occurs df times is $f(N) - f(df)$.[5] A continuous approximation of f is the logarithm to the base 2. The ranking algorithm is displayed in figure 2.10. The weight $\log(N/df)$ will be called the "inverse document frequency": *idf* for short.

$$
\begin{aligned}
\text{vector product:} \quad & \text{score}(\vec{d}, \vec{q}) \; = \; \sum_{k=1}^{n} d_k \cdot q_k \\[2mm]
\text{document term weight:} \quad & d_k = 1 \\[2mm]
\text{query term weight:} \quad & q_k = \log \frac{N}{df}
\end{aligned}
$$

Figure 2.10: *idf* weighting

2.4.2   Probabilistic weighting

The probabilistic model, introduced in section 2.3.3, suggests a simple term weighting algorithm that uses binary document weights and 'relevance weights' for query terms, shown in figure 2.11. Again, *df* is the document frequency

---

[5]The adding of 1 used by Sparck-Jones (1972) was ignored because it is no longer used in later papers (e.g. Robertson and Sparck-Jones 1976).

(n in section 2.3.3), N is the number of documents in the collection, R is the number of known relevant documents and r is the number of relevant documents indexed with the term at hand. This weighting algorithm differs from the other weighting algorithms presented in section 2.4, in that it is derived from the probabilistic model (Robertson and Sparck-Jones 1976).

$$
\begin{aligned}
\text{vector product:} \quad & \text{score}(\vec{d}, \vec{q}) \;=\; \sum_{k=1}^{m} d_k \cdot q_k \\[2mm]
\text{document term weight:} \quad & d_k = 1 \\[2mm]
\text{query term weight:} \quad & q_k = \log \frac{r\,(N - df - R + r)}{(R - r)(df - r)}
\end{aligned}
$$

Figure 2.11: Binary independence weights

2.4.3 $tf.idf$ weighting in the Smart system

The original Smart retrieval system was developed at Harvard University in the early 1960's and later developed at Cornell University. Salton and Yang (1973) experimented with weighting algorithms that use the inverse document frequency. They suggested to combine it with the frequency of a term within a document, the term frequency, tf for short. The introduction of the so-called $tf.idf$ weights is one of the major break-throughs of term weighting in information retrieval. Most modern weighting algorithms are versions of the family of $tf.idf$ weighting algorithms. Salton's original $tf.idf$ weights perform relatively poor, in some cases even poorer than simple idf weighting (see chapter 5).

$$
\begin{aligned}
\text{cosine:} \quad & \text{score}(\vec{d}, \vec{q}) \;=\; \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m} (d_k)^2} \cdot \sqrt{\sum_{k=1}^{m} (q_k)^2}} \\[3mm]
\text{term weights:} \quad & d_k = q_k = tf \cdot \log \frac{N}{df}
\end{aligned}
$$

Figure 2.12: Original $tf.idf$ with cosine normalisation (tfc.tfc)

In 1988, Salton and Buckley summarised the results of 20 years of research into term weighting with the Smart system. A total of 1800 different combinations of term weight assignments were used experimentally, of which 287 were found to be distinct. Experimental results of these term weighting algorithms on

6 document collections were reported. Term weighting algorithms were named by three letter combinations. The first letter indicated the *tf* component, the second component indicates the *idf* component and the third component indicates the normalisation component. For instance, the three letter code tfc is the code for the original *tf.idf* weights with cosine normalisation introduced above. They concluded that the best performing algorithm is one that maps the document vectors differently in the vector space than the query vectors. Figure 2.13 displays the tfc.nfc formula which uses a normalised *tf* factor for the query term weights.

$$
\begin{aligned}
\text{cosine:} \quad \text{score}(\vec{d}, \vec{q}) \; &= \; \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m}(d_k)^2} \cdot \sqrt{\sum_{k=1}^{m}(q_k)^2}} \\[2ex]
\text{document term weight:} \quad d_k &= tf \cdot \log \frac{N}{df} \\[1ex]
\text{query term weight:} \quad q_k \; &= \; (\, 0.5 \; + \; \frac{0.5 \, tf}{\max tf} \,) \cdot \log \frac{N}{df}
\end{aligned}
$$

Figure 2.13: tfc.nfc term weighting algorithm

The start of the TREC conferences in 1992 gave a new impulse to term weighting experiments. An important discovery is that weights that are logarithmic in *tf* outperform weighting algorithms that are linear in *tf*. Buckley, Allan, and Salton (1994) suggest to use the algorithm of figure 2.14 which is called the lnc.ltc formula, where the 'l' stands for weights with a logarithmic *tf* component.

$$
\begin{aligned}
\text{cosine:} \quad \text{score}(\vec{d}, \vec{q}) \; &= \; \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m}(d_k)^2} \cdot \sqrt{\sum_{k=1}^{m}(q_k)^2}} \\[2ex]
\text{document term weight:} \quad d_k &= 1 + \log(tf) \\[1ex]
\text{query term weight:} \quad q_k &= (1 + \log(tf)) \cdot \log \frac{N+1}{df}
\end{aligned}
$$

Figure 2.14: lnc.ltc term weighting algorithm

One of the recent weighting algorithms Lnu.ltu uses a combination of the document length and the average document length instead of the cosine measure for length normalisation. The algorithm outperforms the cosine versions on the TREC collections, but lacks the nice metaphor of measuring the angle between

two vectors in a Euclidean space (Singhal, Buckley, and Mitra 1996).

$$
\begin{aligned}
\text{vector product:} \quad &\text{score}(\vec{d}, \vec{q}) \;=\; \sum_{k=1}^{m} d_k \cdot q_k \\[2mm]
\text{document term weight:} \quad &d_k = \mathbf{L} \times \mathbf{u} \quad (\mathbf{Lnu}) \\
\text{query term weight:} \quad &q_k = \mathbf{l} \times \mathbf{t} \times \mathbf{u} \quad (\mathbf{ltu}) \\[2mm]
\text{tf factors:} \quad &\mathbf{l} = 1 + \log(\mathit{tf}) \\
&\mathbf{L} = \frac{1 + \log(\mathit{tf})}{1 + \log(\text{average } \mathit{tf} \text{ in document})} \\[2mm]
\text{idf factor:} \quad &\mathbf{t} = \log\left(\frac{N+1}{\mathit{df}}\right) \\[2mm]
\text{length norm. factor:} \quad &\mathbf{u} = \frac{1}{(1-s) + s\,\frac{\text{number of unique words in text}}{\text{average number of unique words}}}
\end{aligned}
$$

Figure 2.15: Lnu.ltu algorithm

2.4.4   Linear combinations of relevance clues

By the late 1980's and early 1990's, researchers had a pretty good idea which information is important for good performing term weighting algorithms. Fuhr and Buckley (1991) therefore suggests the following approach. Based on experience with term weighting algorithms, develop a function that is a linear combination of clues that are good indicators of the document's relevance given a query term. Then, use a retrieval test collection with corresponding queries and relevance judgements to fit the function to the data. An example of such a function of linear clues, is the following by Gey (1994).

$$
\begin{aligned}
w \;=\;\; & c_0 \;+\; c_1 \log \mathit{qtf} \;+\; c_2 \log \frac{\mathit{qtf}}{\text{query length}} \;+\; c_3 \log \mathit{tf} \\
& +\, c_4 \log \frac{\mathit{tf}}{\text{document length}} \;+\; c_5 \log \frac{N}{\mathit{df}} \;+\; c_6 \log \frac{\text{collection length}}{\mathit{cf}}
\end{aligned}
$$

In the formula cf stands for 'collection frequency': the number of occurrences of a term in the collection. Fuhr and Buckley (1991) used polynomial regression to estimate the values of $c_0, c_1, \cdots$. Alternatively, the method of logistic regression might result in better results (Gey 1994).

2.4.5   Term weighting in the Inquery system

The Inquery system was developed at the University of Massachusetts, Amherst in the late 1980's. The system uses the inference network approach introduced in section 2.3.8. Turtle and Croft (1991) report on term weighting experiments

with a similar linear combination of retrieval clues as in section 2.4.4, one of the form $\alpha + \beta \cdot tf + \gamma \cdot idf + \delta \cdot tf \cdot idf$. The best performance was achieved when $\alpha = 0.4$, $\beta = \gamma = 0$ and $\delta = 0.6$. Variations that work about as well use a logarithmic normalisation for the *tf* component. Figure 2.16 shows the ranking algorithm that was used in the first years of TREC (Broglio et al. 1995). The document term weights are not zero if the term does not occur in the document, but instead take the default value $\alpha = 0.4$. The parameter b determines the effect of the penalty for long documents and w_q is a weight given for to the whole query. In later versions of the Inquery system, the Okapi's non-linear *tf* function was used, which is explained below.

$$
\begin{array}{rl}
\text{Inquery weighted sum:} & \text{score}(\vec{d}, \vec{q}) \;=\; \dfrac{w_q \cdot \sum_{k=1}^{m} d_k \cdot q_k}{\sum_{k=1}^{m} q_k} \\[2.5em]
\text{document term weight:} & d_k = 0.4 + 0.6 \cdot (b\, H + (1-b)\, ntf\,) \cdot nidf \\
\text{query term weight:} & q_k = tf \\[1.5em]
\text{penalty long documents:} & H = \left\{ \begin{array}{ll} 1.0 & \text{if } \max tf \le 200 \\[0.5em] \frac{200}{\max tf} & \text{otherwise} \end{array} \right. \\[2.5em]
\text{normalised tf:} & ntf = \dfrac{\log(tf + 0.5)}{\log\,(\max tf + 1.0)} \\[2.5em]
\text{normalised idf:} & nidf = \dfrac{\log\,(N/df)}{\log N}
\end{array}
$$

Figure 2.16: Inquery weighting algorithm

2.4.6   Term weighting in the Okapi system

The Okapi system was originally developed at the Polytechnic of Central London in the early 1980's and later developed at City University London and Microsoft Research. The system is based on the probabilistic model introduced in section 2.3.3. Because the performance of the probabilistic weighting was poor on TREC-1, Robertson and Walker (1994) experimented with weighting algorithms that take the term frequency and document length into account. They tried a number of weighting algorithms which led to the BM25 algorithm (BM stands for best match) presented in figure 2.17. The algorithm uses weights that are approximately linear for small values of *tf*, but do not increase in the same rate for larger values of *tf*, similar to the algorithms of figure 2.14 and figure 2.16 that use $\log(tf)$. This behaviour is suggested by the behaviour of equations 2.17 and 2.18 in section 2.3.7 that combines the 2-Poisson weighting with the probabilistic model. The parameters k_1 and k_3 determine the rate in

which the weights increase with tf. The parameter b determines the effect of the document length normalisation component (Robertson et al. 1999).

$$
\begin{aligned}
\text{vector product:} \quad & \text{score}(\vec{d}, \vec{q}) \;=\; \sum_{k=1}^{m} d_k \cdot q_k \\[2mm]
\text{document term weight:} \quad & d_k = \frac{(k_1 + 1)\; tf}{K + tf} \\[2mm]
\text{query term weight:} \quad & q_k = \frac{(k_3 + 1)\; tf}{k_3 + tf}\; w \\[2mm]
\text{length normalisation:} \quad & K = k_1\!\left((1-b) + b\;\frac{\text{document length}}{\text{average doc. length}}\right) \\[2mm]
\text{relevance weight:} \quad & w = \log \frac{(r + 0.5)(N - df - R + r + 0.5)}{(R - r + 0.5)(df - r + 0.5)}
\end{aligned}
$$

Figure 2.17: Okapi BM25 algorithm

2.5 Discussion

This chapter summarises over thirty years of research into ranking algorithms for information retrieval by presenting the most influential models and weighting algorithms in the field. The selection made in this chapter was based on models that attempt to solve one of the following three problems:

1. term weighting and ranking algorithms;
2. relevance feedback from examples of relevant documents;
3. structured queries and the ability to combine information.

None of the existent models of information retrieval address the three problems at the same time. Three models try to unify term weighting and ranking algorithms, without the use of one of the ad-hoc term weighting algorithms presented in section 2.4: the probabilistic model, the 2-Poisson model and the combination of the two. The former is too simple to reach high retrieval performance and the latter two are too complex to make reliable parameter estimation possible. The combination of the probabilistic model and 2-Poisson model, however, inspired the BM25 term weighting algorithm. The vector space model and the probabilistic model account for relevance feedback. The former intuitively by Rocchio's algorithm, doing query term reweighting and query expansion, and the latter more formally grounded in the model, doing query term reweighting but no query expansion. The fuzzy set models, the p-norm model and the Bayesian network models account for the use of structured queries. The Bayesian network

model also accounts for the combination of evidence from different sources, for instance from controlled terms and from free text.

The section gives the background of the first research question this thesis tries to answer: *How to apply the theory of statistical language models to three classical problems of matching models of information retrieval: term weighting, relevance feedback and structured queries?* None of the existent models of information retrieval address the three problems at the same time. Of course, many retrieval systems support term weighting, feedback and structured queries, but they are either based on ad-hoc considerations or on more than one of the models introduced above. The information retrieval language models will address the three problems into one unifying theory in section 4.

Chapter 3

Today's information retrieval systems in practice

This chapter lists some of the features and search capabilities that can be found in today's experimental and commercial full text information retrieval systems. The chapter is divided in two sections, the first on automatic query systems, and the second on query operators for manual query formulation.

3.1 Introduction

A model of the query formulation process should formalise two things. Firstly, the selection of query terms and secondly, the selection of query operators. This chapter describes many of the practical approaches to term selection and operator selection. Query term selection is described in section 3.2. The section describes a number of simple but effective approaches to select terms automatically from a user request, for instance stop word removal and stemming. Practical query operators are described in section 3.3. In practice, query operators are almost exclusively used during *manual* query formulation. The new retrieval model suggested in the next chapter should support the use of query operators in a manual query formulation process as well. Examples of these operators are proximity operators, and mandatory term operators.

3.2 Automatic query systems

With the emergence in the 1970's of models of ranked retrieval that process unstructured queries, automatic query systems became a fact. The main philosophy of automatic query systems is that indexing and query formulation should result in a representation that is closer to the actual meaning of the text, ignoring as many of the irregularities of natural language as possible. A typical approach to indexing and query formulation selects the query terms as

37

follows. First a tokenisation process takes place, then stop words are removed, and finally the remaining words are stemmed. Additionally, natural language processing modules might provide the identification of phrases or splitting of compounds. Figure 3.1 shows an example text that will be used to illustrate the typical approach to query term selection.

CHAPTER 1
PREAMBLE

1.1. Humanity stands at a defining moment in history. We are confronted with a perpetuation of disparities between and within nations, a worsening of poverty, hunger, ill health and illiteracy, and the continuing deterioration of the ecosystems on which we depend for our well-being.

Figure 3.1: An example text: the opening lines of Agenda 21

3.2.1   Tokenisation

As a first step in processing a document or a query, it has to be determined what the processing tokens are. One of the most simple approaches to tokenisation defines word symbols and inter-word symbols. In the example of figure 3.2 all characters that are no letter and no digit are considered to be inter-word symbols. The inter-word symbols are ignored during this phase, and the remaining sequences of word symbols are the processing tokens. As a result it is not possible to search for punctuation marks like for instance hyphens and question marks.

```
chapter 1 preamble 1 1 humanity stands at a defining moment in
history we are confronted with a perpetuation of disparities
between and within nations a worsening of poverty hunger ill
health and illiteracy and the continuing deterioration of the
ecosystems on which we depend for our well being
```

Figure 3.2: The Agenda 21 text after tokenisation

In the example, mark-up information is also ignored, but this information might be kept to search for e.g. title words. Heuristics might be used to identify sentences, or the fact that "1.1" should be kept as one processing token. The basic tokenisation process may be enhanced by treating multiple sequences of word symbols as one token or by splitting one sequence of word symbols into two or more tokens. Some of these approaches are addressed further in section 3.2.4 and 3.2.5.

3.2.2   Stop word removal

Stop words are words with little meaning that are removed from the index and the query. Words might carry little meaning from a frequency (or information theoretic) point of view, or alternatively from a linguistic point of view. Words that occur in many of the documents in the collection carry little meaning from a frequency point of view. They get a low weight because of the *idf* component in the weighting algorithms of section 2.4. By removing the very frequent words, the document scores will not be affected that much. Stop word removal on the basis of their frequency can be done easily by removing the 200-300 words with the highest collection frequencies. As a result of stopping the very frequent words, indexes will be between 30 % and 50 % smaller (Schäuble 1997).

If words carry little meaning from a linguistic point of view, they might be removed whether their frequency in the collection is high or low. In fact, they should especially be removed if their frequency is low, because these words affect document scores the most. Removing stop words for linguistic reasons can be done by using a stop list that enumerates all words with little meaning, like for instance "the", "it" and "a". These words do also have a high frequency in English, but most publicly available stop lists are, at least partly, constructed from a linguistic point of view. For instance the stop list published by Van Rijsbergen (1979), contains words like "hereupon" and "whereafter", which occur respectively two and four times in the TREC-8 collection and never in for instance the Cranfield collection. Stop lists are used in many systems, but the lengths of the various stop lists may vary considerably. For instance, the Smart stop list contains 571 words (Smart 1994), whereas the Okapi system uses a moderate stop list of about 220 words (Robertson and Walker 2000).

```
chapter 1 preamble 1 1 humanity stands defining moment
history confronted perpetuation disparities nations
worsening poverty hunger ill health illiteracy continuing
deterioration ecosystems depend well being
```

Figure 3.3: The Agenda 21 text after removing words from the Smart stop list

In section 4, stop words are defined mathematically by assigning zero probability to one of the model's parameters. The mathematical definition does not conflict with the linguistically motivated definition of stop words.

3.2.3   Morphological normalisation

Morphological normalisation of words in documents and queries is used to find documents that contain morphological variants of the original query. Morphological normalisation can be achieved either by using a stemmer or by using dictionary lookup.

A stemmer applies morphological 'rules of the thumb' to normalise words. Stemmers were already developed in the 1960's when the first retrieval systems were implemented. Well known stemmers are those by Lovins (1968) and Porter (1980), the last one being the most commonly accepted algorithm. As reported by Harman (1991) for English and (Kraaij and Pohlmann 1996) for Dutch, the effect on retrieval performance is limited. Stemming tends to help as many queries as it hurts. Sometimes stemming algorithms may conflate two words with very different meanings to the same stem, for instance the words "skies" and "ski" will both be reduced to "ski". In such cases users might not understand why a certain document is retrieved and may begin to question the integrity of the system in general (Kowalski 1997). Still, stemmers are used often in many research systems like Smart, Okapi and Twenty-One. The Inquery system uses a stemming technique called Kstem that combines dictionary lookup and stemming rules (Broglio et al. 1994). Figure 3.4 gives the results of the Porter algorithm, which does not always result in linguistically correct stems.

```
chapter 1 preambl 1 1 human stand defin moment
histori confront perpetu dispar nation worsen
poverti hunger ill health illiteraci continu
deterior ecosystem depend well be
```

Figure 3.4: The Agenda 21 text after stemming

Dictionary lookup will result in linguistically correct stems, often called *lemmas*. Having a full-form dictionary is however not enough to build a lemmatiser. Some words will have multiple entries, possibly with different lemmas. For instance, the word "saw" may be a past tense verb, in which case its lemma is "see" and it may be a noun, in which case its lemma is equal to the full form. Another example is the word "number" which may be the comparative of "numb". For these cases, a lemmatiser has to determine the word's part-of-speech before the correct lemma can be chosen. Statistical algorithms trained on (partially) hand-tagged corpora may be used to effectively find the correct part-of-speech and therefore the correct lemma.

3.2.4   Phrase extraction

During indexing and automatic query formulation, multiple words may be treated as one processing token. The meaning of phrases might be quite different from the meaning of the separate words. A user who enters the query "stock exchange" will probably not be satisfied with documents that discuss "exchange of live stock". There are three basic approaches to phrase extraction. Phrases might be simply predefined (Robertson and Walker 2000), extracted by statistical co-occurrence (Mitra et al. 1997) or extracted by syntactic processing

(Strzalkowski 1995). Phrase extraction based on statistical co-occurrence may use very simple methods, e.g. the identification of all pairs of non stop words that occur contiguously in at least X documents. Syntactic processing might be used to extract noun phrases which are then normalised to head-modifier pairs. This will produce the same processing token for e.g. "information retrieval" and "retrieval of information", because in both "information" modifies the head "retrieval". Statistical and syntactic techniques for phrase extraction were compared by Mitra et al. (1997) for English and Kraaij and Pohlmann (1998) for Dutch. Both evaluations show that phrase extraction, like stemming, does not improve retrieval effectiveness significantly. The most successful methods use both the phrase and the single words in the index.

The phrase and its single words are obviously related, because the occurrence of the phrase implies the occurrence of its single words. The application of ranking algorithms that use term independence might therefore no longer by justified. This complication is not addressed by the publications mentioned above, but in fact, the obvious violation of the independence assumption might be one of the reasons for the disappointing results on retrieval performance. In section 4.8 a bigram model will be introduced that explicitly models the dependence relation between words in phrases.

3.2.5 Compound splitting

During indexing or query formulation, some words might be treated as more than one processing token. A compound word is a single orthographic unit that consists of two or more single words, like for instance "airport" and "wildlife" (Allan et al. 2000). Compound words are especially an issue in languages that allow almost unrestricted compounding like Dutch and German. In Dutch, for instance the noun phrase "potable water supply" would be one compounded word: "drinkwatervoorziening". Unfortunately, compound splitting might result in accidental splitting of proper names and other words that are not listed in the dictionary, for instance "Washington" is not the composition of the German words "Was", "hing" and "Ton" (Schäuble 1997). Kraaij and Pohlmann (1998) show that the splitting of compounds improves retrieval performance significantly for Dutch. Similar to phrases, both the compound and its components can be used during searching, but the use of a retrieval model that assumes the independence between terms might not be appropriate.

3.2.6 Synonym normalisation

Like stemming and lemmatisation, synonymous words might also be conflated to one processing token during indexing and automatic query formulation. For instance in Okapi, closely related or synonymous terms like "CIA" and "Central Intelligence Agency" are conflated (Robertson and Walker 2000). In Inquery special processing tokens like `#CITY` and `#COMPANY` are added for respectively every mention of a U.S. city or company (Broglio, Callan, and Croft 1994).

3.3   Operators for manual query formulation

Despite the existence of the automatic query systems described above, Boolean retrieval had a monopoly in the world of commercial information retrieval systems for almost three decades. In the mid 1990's the monopoly was finally broken when the major database vendors like e.g. Dialog and Lexis-Nexis added natural language search functionality to their systems: Dialog offered Target and Lexis-Nexis offered FreeStyle (Brenner 1996). In these systems, natural language searching is not intended to replace Boolean searching, but instead is added as an auxiliary module. At the same time web search engines like Hotbot (1995) an AltaVista (1996)[1] were launched that offered simple natural language search. Like the commercial database vendors, these engines offer the good old Boolean retrieval via their advanced search options. The practical use of Boolean operators, and extensions of the Boolean model for proximity searching are described in section 3.3.1 and section 3.3.2. Section 3.3.3 describes the use of wildcards. Section 3.3.4 describes the new natural language search facilities of today's commercial information retrieval systems. Section 3.3 is based on similar overviews of Kowalski (1997), Chowdhury (1998) and Rasmussen (1999).

3.3.1   Standard Boolean operators: AND, OR, NOT

The Boolean model and its operators were introduced in section 2.2. This section describes the model's practical use. Expert users of traditional Boolean retrieval systems tend to use faceted queries (Kekäläinen 1999). A faceted query is a query that uses disjuncts of quasi-synonyms: the facets, conjoined with the AND operator. The following query for instance has two facets: `(biotechnology OR biological resources) AND (human health OR malnutrition OR poverty)`. If documents are indexed manually by a documentalist, the query retrieves documents *about* the two facets, not necessarily containing any of the exact words. Automatic full text indexing usually does not do much more than identifying words and putting them all in the index. Therefore, if documents are indexed automatically by their full text, the Boolean operators get a slightly different purpose (Salton, Fox, and Wu 1983). The AND operator may be used to identify phrases as in `biological AND resources`. The OR operator may relate synonymous terms as in `poverty OR hardship OR destitution OR indigence`, which might be necessary because `poverty` is no longer a controlled term. The use of the AND operator for phrases and the OR operator for real synonyms is not really an issue in systems that use manual indexing with controlled terms, because phrases are precoordinated and synonyms are explicitly avoided by the documentalist.

   Usually systems have a default order in which the Boolean operators are processed, either from left to right or possibly with precedence of AND over OR as in SQL. Parentheses can be used to specify a different order than the default. The NOT operator is usually implemented as AND NOT.

---

[1] Web addresses are listed in the bibliography and cited following the convention (name year), where the year is the year that the site was launched.

query	interpretation
`(renewable OR sustainable) AND development`	select documents containing the term "development" and one or both of the terms "renewable" and "sustainable"
`renewable OR (sustainable AND development)`	select documents containing either the term "renewable" or both the terms "sustainable" and "development"
`development NOT sustainable`	select documents that contain the term "development", that do not contain the term "sustainable"

Table 3.1: Standard Boolean operators

3.3.2 Proximity searching: ADJ, NEAR

With the emergence of automatic full text indexing, commercial retrieval systems added new Boolean operators to the standard Boolean operators mentioned in section 3.3.1. These operators use the positions of words in text to compensate for the loss of expressiveness caused by using separate words instead of complex manual index terms. The ADJ operator allows for the search of exact phrases by looking for documents that contain two adjacent terms in the specified order, for instance `environmental ADJ damage` selects only documents containing the exact phrase "environmental damage". The NEAR operator allows for the search of two terms that are near to each other without any requirements on the order of the words. Table 3.2 list some examples.

query	interpretation
`waste ADJ management`	select documents containing the exact phrase "waste management"
`waste NEAR management`	select documents containing e.g. "waste management", "management of waste" or "waste of valuable management talent"
`(hazardous OR toxic) ADJ wastes`	select documents containing either "hazardous wastes" or "toxic wastes"
`(hazardous AND waste) ADJ management`	ill-defined because "management" could not be adjacent to both "hazardous" and "waste"

Table 3.2: Proximity operators

In the traditional Boolean model, single terms and Boolean combinations of

terms are represented by sets of documents as presented in section 2.2. With
the introduction of proximity searching, the set of a single term should some-
how include the positions of the term in the document. If two single terms
are combined by the Boolean OR, then the result set still includes the position
information, because any occurrence of `a` or `b` has its own position in a docu-
ment. However, if two single terms are combined by the Boolean AND, then
the result set no longer includes useful position information, because there are
no actual positions on which `a` and `b` occur in a document. Some combina-
tions of operators might therefore be ill-defined, for instance the combination
of an AND-result set with the ADJ operator as shown in the last row of table
3.2. Some systems produce an error if such a query is entered, but usually
system designers decide to process the ill-defined example of table 3.2 as e.g.
`(hazardous ADJ management) AND (waste ADJ management)`.

3.3.3   Wildcards

Wildcards are used to mask part of a query term with a special character, allow-
ing it to match any term that maps to the unmasked portion of the query term.
Table 3.3 shows some examples of the use of wildcards, taken from Kowalski
(1997). Of the options in table 3.3, suffix searches are the most common. In

query	interpretation
`dog*`	suffix truncation selects documents containing e.g. "dog", "dogs" or "doggy", but also "dogma" and "dogger"
`*computer`	prefix truncation: selects documents containing e.g. "minicomputer", "microcomputer" or "computer"
`colo*r`	infix truncation: selects documents containing e.g. "colour", "color", but also "colorimeter" or "colourbearer"
`multi$national`	single position truncation: selects documents containing "multi-national" or "multinational", but no "multi national" if it is two processing tokens

Table 3.3: Wildcards

some systems suffix searches are the default without the user having to specify
this. Suffix truncation is also the easiest of the options above to implement.
Term lookup is often implemented by sorting the index terms in alphabetic or-
der or by using a trie. Prefix truncated terms cannot use the alphabetically
sorted term list or the trie and therefore should use a linear search through the
entire list. One possible way to support fast lookup is to include all possible
rotated word forms in the list (Salton 1989).

In the Boolean model, wildcards are nicely defined by assuming an OR operator. Searching for `dog*` is like searching for `dog OR dogs OR doggy OR ...` Because the OR operator does not conflict with position information, the use of term expansion is well-defined if used in combination with the NEAR and ADJ operators. There is no obvious way to define wildcards in the models of ranked retrieval presented in section 2.3.

3.3.4 Natural language search

Usually, 'natural language search' means that the user only has to type a request and the system takes care of automatic query formulation. It seems therefore strange to write about natural language search in a section on query operators and *manual* query formulation. Strictly speaking this is right, but natural language search is for many commercial systems synonymous with search facilities with ranking capabilities that do not require knowledge of Boolean set operators. Instead of the Boolean operators, these systems often use related operators that are more easy to understand by non-expert users. The use of these operators is however not mandatory, making it possible to enter a request as shown in the first example of table 3.4. The operators, which can be found in for instance Dialog Target, Lexis-Nexis FreeStyle, or Altavista are summed up in the following paragraphs. In these systems, the actual tokens used for these operators might differ from the ones used in the examples. Section 3.2 addresses the strict interpretation of natural language searching that does not require operators at all.

query	interpretation
`how to promote sustainable consumption patterns`	real natural language request: rank the documents containing one or more of the terms
`how to reduce the production of +"harmful materials" -uranium`	rank the documents considering that documents should contain the phrase "harmful materials" and should not contain that the term "uranium'
`why (forbid prohibit ban) wasteful packaging[0.9] of products[0.1]`	rank the documents considering that the terms "forbid", "prohibit" and "ban" are synonyms and considering that "packaging" is much more important than "products"

Table 3.4: Natural language search operators

Exact match operator / mandatory terms

The mandatory term operator can be used to indicate that a term *must* be present in the selected documents. It is inspired by the AND operator in Boolean queries, but has slightly different semantics. Unlike the AND operator, which is a binary operator requiring two arguments, the mandatory term operator is a unary operator requiring one argument. The example uses the plus symbol to flag mandatory terms, but other conventions are also used, e.g. using the asterisk-character, or using a separate user interface field.

Exclusion operator

The exclusion operator can be used to indicate that a term should *not* be present in the selected documents. Obviously it is inspired by the NOT operator in Boolean queries. This operator is not as common as the exact match operator, because the absence of a term is not as clear an indication of relevance as the presence of a term. In table 3.4 the minus symbol is used to flag terms that documents should not contain.

Synonyms and wildcards

Operators for synonyms and wildcards are inspired by the Boolean OR. Usually the operator for wildcards used for natural language queries does not differ from the operator used in Boolean queries. Explicit marking of synonyms is sometimes supported by putting synonyms between parenthesis. The system uses this information to produce a better ranking. Other conventions leave the main term outside the parenthesis as in `child (minor, infant)`.

Phrases

Explicit marking of phrases is inspired by the Boolean ADJ operator. The system uses the phrase to produce a better ranking. Identifying phrases is very useful in combination with the exact match operator to perform a high-precision search, looking for an exact phrase or an exact quotation. Most query languages use single or double quotation marks to mark phrases.

Manual term weighting

Query term weights are, one way or the other, used in many ranking algorithms. Some systems give the user access to these weights so they can indicate themselves which terms are important and which terms are not important. The last row in table 3.4 gives an example of this use of term weights. Manual term weighting is for instance supported by Microsoft Index Server.

3.3.5   Field search

Although documents in a retrieval system might just be represented by their text only, this is generally not sufficient for professional applications. A quite

standard, but very important feature of information retrieval systems is the support of different fields per document. If this is the case, the document is usually called (database) record. Some of these fields, like for instance the 'title' field or the 'abstract' field might be treated as full text fields. Others, like for instance 'publication date' or 'language' might be treated as predefined, structured data as in traditional database management systems. Each application might have its own domain dependent fields. For instance, in web search engines there might be a 'url' field that stores the web address (uniform resource locator) of the document.

Users may want to restrict their full text search to documents that were for instance published during the last decade, or the last four weeks. Also, users may search fields separately, for instance to retrieve all documents with the words "harmful" and "materials" in the title. The last example has a by now familiar problem if it is used in a Boolean retrieval system. If there are no documents with either the word "harmful" or the word "materials" in the title, then the system will not retrieve anything. In this case however, the user might still be helped if the system retrieves documents with one or both words in the abstract or in the full text. In fact, the user might prefer a document that contains both words in the abstract over a document that contains only one of the words in the title.

3.4 Discussion

This chapter provided some background to the following research question: *How to apply the theory of statistical language models to the automatic formulation of structured queries from natural language search statements?* Section 3.2 presented a number of techniques for automatic indexing and automatic query formulation that have been extensively studied by the information retrieval research community. Two of these techniques, the use of stop words and stemmers, are standard practice in research systems, and to a lesser extend also in commercial systems. In terms of retrieval models, there has been little attention to stop words and stemmers. For instance, from the viewpoint of the vector space model, there is no good reason why one should remove certain words, and there is certainly no reason to conflate words to a common stem since this violates the orthogonality of vectors. Traditionally this was never a problem, because the indexing and query formulation processes were considered to fall outside the scope of the mathematical models of information retrieval. This thesis tries to break with this tradition by presenting an explicit model of the query formulation process. The new model of information retrieval that is introduced in the next chapter integrates a query formulation model and a matching model, and gives for instance a mathematical interpretation of stemming.

The automatic formulation of structured queries might benefit from a model that explains or defines the advanced query operators for free text presented in section 3.3. A number of these query operators are not covered by any of the models of ranked information retrieval. Examples of these facilities are

wildcards, proximity operators, synonym operators, mandatory query term operators or a text search that is restricted to title words only. The language model-based retrieval system presented in chapter 4 suggests ranked retrieval versions of the proximity operators, and can be used to define or explain the other operators introduced in this chapter.

Chapter 4

A language model-based information retrieval system

This chapter presents a new probabilistic model of information retrieval based on the use of statistical language models. Section 4.1 introduces the term 'languages models', gives a summary of related research and informally describes this approach to information retrieval. Section 4.2 formally introduces the basic model. In section 4.3 translation of terms is added to model the automatic formulation of structured queries from a natural language search statement. Section 4.4 addresses the notion of importance of query terms. Section 4.5 and 4.6 will present the exact same models in terms of respectively hidden Markov models and Bayesian networks. Section 4.7 addresses implementation details and shows the resemblance of the resulting weighting formulas with other models. Finally, section 4.8 introduces extensions for proximity searching.

4.1 Introduction

4.1.1 A short history of language models

Statistical language models have been around for quite a long time. They were first applied by Andrei Markov at the beginning of the 20th century to model letter sequences in works of Russian literature (Manning and Schütze 1999). Another famous application of language models are Claude Shannon's models of letter sequences and word sequences, which he used to illustrate the implications of coding and information theory (Shannon 1948). Later, statistical language models were developed as a general natural language processing tool. Language models were first successfully used for automatic speech recognition at the end of the 1970's. The by now standard model of automatic speech recognition consists of two parts. The first part is the language model, that predicts the next word in continuous speech. The second part models the acoustic signal and is therefore called the acoustic model. The theory behind the speech recognition models

is part of hidden Markov model theory (indeed, a 'hidden' version of Markov's models) that was developed by Leonard Baum and his colleagues at IBM in the late 1960s and early 1970s (Rabiner 1990; Jelinek 1997). Recently, hidden Markov models are studied as part of a general graphical model formalism, which subsumes many of the multivariate probabilistic models used in statistics, systems engineering, information theory and pattern recognition. Examples include Bayesian networks, Markov random fields, factor analysis and Kalman filters (Jordan 1998; Bengio 1999).

4.1.2   The application to information retrieval

Only very recently, since 1998, statistical language models are applied to information retrieval. The past two years show a remarkably large number of publications in which statistical language models are used to compute the ranking of documents given a query. To sum them up quickly: Ponte and Croft (1998) were the first to suggest the use of language models in information retrieval. They used estimation based on risk functions to overcome the problem of small sample sizes. Hiemstra (1998a) and Hiemstra and Kraaij (1999) were the first to introduce ranking based on a mixture of global and local probability distributions that is also used in the publications mentioned in the remainder of this paragraph. Miller, Leek, and Schwartz (1999) use hidden Markov models for ranking, including the use of bi-grams to model two word phrases and a method for performing blind feedback. Sahami (1999) suggested an approach to document clustering based on smoothing the document models by using the geometric mean of the global and local distributions. Berger and Lafferty (1999) and Hiemstra and De Jong (1999) developed a model that includes statistical translation. Ng (2000) introduced a model that uses the ratio of the conditional probability of the query given the document and the prior probability of the query, including a method for query expansion. Song and Croft (1999) used a model which includes bi-grams and introduced Good Turing re-estimation to smooth the document models. This chapter will address details of many of the above mentioned publications. They will be recited where appropriate in the following sections. It is assumed that the reader is familiar with the basics of probability theory as for instance presented by Mood and Graybill (1963).

4.1.3   Two models of information retrieval processes

This chapter will introduce two models of information retrieval: a basic retrieval model and an extension of the basic model, the statistical translation retrieval model. The basic model defines the system's matching process. It has the same function as the models presented in chapter 2. The extended model adds statistical translation to the basic retrieval model to model both the matching process and the query formulation process. Because today's computers are still not able to really understand the documents and the user's request, both matching and query formulation are modelled by simple probability mechanisms. Matching is modelled by the generation of a random query from a relevant document

and query formulation is modelled by translation of the query into the request (Hiemstra and De Jong 1999).
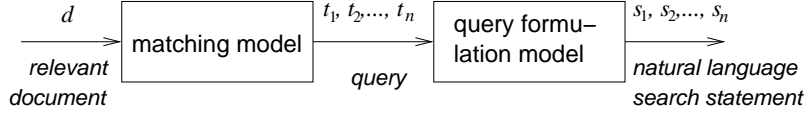
Figure 4.1: Model of matching and query formulation

Figure 4.1 suggests an information theoretic view of the problem (Miller, Leek, and Schwartz 1999; Berger and Lafferty 1999). Information theory was developed by Shannon (1948) to model the problem of decoding a message that is sent over a noisy communication channel. From this viewpoint, a relevant document d gets 'corrupted' into a query $t_1, \cdots, t_n$ by sending it through a noisy channel, and the query gets again corrupted into a request $s_1, \cdots, s_n$ by sending it through a second noisy channel. A natural language information retrieval system can be thought of as a decoding function $f : s_1, \cdots, s_n \rightarrow d$, that tries to reproduce the message that was originally sent, that is, to find the document that is relevant to the request. An optimal retrieval system will choose $f(s_1, \cdots, s_n)$ such that:

$$f(s_1, \cdots, s_n) \quad = \quad \underset{d}{\text{argmax}} \ P(D{=}d|S_1{=}s_1, \cdots, S_n{=}s_n)$$

By Bayes' rule and because $P(S_1 = s_1, \cdots, S_n{=}s_n)$ does not depend on d:

$$= \quad \underset{d}{\text{argmax}} \ P(S_1{=}s_1, \cdots, S_n{=}s_n, \ D{=}d)$$

$$= \quad \underset{d}{\text{argmax}} \ \sum_{t_1, \cdots, t_n} P(S_1{=}s_1, \cdots, S_n{=}s_n, \ T_1{=}t_1, \cdots, T_n{=}t_n, \ D{=}d)$$

Because there are two independent channels:

$$= \quad \underset{d}{\text{argmax}} \ \sum_{t_1, \cdots, t_n} P(S_1 = s_1, \cdots, S_n{=}s_n|T_1 = t_1, \cdots, T_n{=}t_n)$$
$$P(T_1 = t_1, \cdots, T_n{=}t_n|D{=}d)P(D{=}d)$$

$P(D{=}d)$ is the prior probability of relevance of the document d, and $P(T_1 = t_1, \cdots, T_n{=}t_n|D{=}d)$ is the probability of the query given a relevant document. Together, $P(D = d)$ and $P(T_1 = t_1, \cdots, T_n = t_n|D = d)$ define the matching model. $P(S_1 = s_1, \cdots S_n = s_n|T_1 = t_1, \cdots, T_n = t_n)$ is the probability of the natural language request given the query, which defines the query formulation

model. A real life retrieval system does not know these probabilities, but instead defines them by some simple basic principles. A basic principle for the matching model might be that each document has the same probability of being relevant, and that within a document each occurrence of a term has the same probability of ending up in the query. A basic principle for the query formulation model might be that each query term is translated to one and only one word in the request.

4.1.4   How the system works

For each document in the collection, a two-step statistical model defines the probability of generating the user request. Documents are ranked according to this probability. If a request is entered, the system first uses the query formulation model to hypothesise for each word in the request the terms that might have generated it. This results in a structured query that represents all queries that might have generated the request. In a second step, the system uses the matching model of each document to calculate the probability that the document generated any of the queries represented by the structured query.

The two parts have objectives that are similar to the two parts of the speech recognition models. The objective of the translation model of information retrieval is similar to the acoustic model of speech recognition. Both model the observed signal, respectively the user's request and the sound wave. The structured query that represents all queries that might have generated the request, can be compared to a so-called word lattice in speech recognition (Rabiner 1990). The objective of the basic model of information retrieval is similar to the objective of the language model of speech recognition. The models predict respectively the next term in the query, and the next word in speech. So, the basic retrieval model is the 'true' language model, and the translation model is the signal model. The distinction between language model and signal model can also be made for e.g. models for part-of-speech tagging (Cutting et al. 1992), and models for statistical machine translation (Brown et al. 1990). The major difference with the models of speech recognition, part-of-speech tagging and machine translation, is that for information retrieval there is a separate language model for each document in the collection.

4.1.5   The query formulation model

For the query formulation model a simple one-to-one statistical translation model will be used (Hiemstra 1998b), that is, each query term is translated to one, and only one request word. The model requires easier calculations during actual use than the one-to-many models of Brown et al. (1990) which are quite standard in the field. Training a one-to-one model from data, for instance training a machine translation lexicon from a parallel corpus, is however less straightforward, but can be done efficiently by some effective approximations. The training of statistical translation models is not addressed by this thesis. The existence of a translation tool for query formulation is simply assumed. Any

natural language processing tool or algorithm that converts natural language words into some other representation may be used as the translation/query formulation tool. Examples are stemming algorithms (Porter 1980), edit distance algorithms (Baeza-Yates 1992), fuzzy matching algorithms (De Heer 1979), the soundex algorithm (Gadd 1988), ontologies as Wordnet (Miller et al. 1990), or machine-readable bilingual dictionaries.

4.1.6 The matching model

The matching model assumes that relevant documents are drawn at random from the document collection. Given a relevant document, queries are generated by the explicit generation of important terms and unimportant terms. The important terms are supposed to be drawn at random from the document. The unimportant terms are supposed to be drawn at random from the full collection. The probabilities of drawing the terms from the document are calculated by a simple procedure that, in introductory courses on probability theory (Mood and Graybill 1963), is often explained by urns containing coloured balls. Consider 4 urns with coloured balls, one of them with 3 red balls, 1 blue ball and 6 yellow balls. For instance, the probability of selecting at random the described urn and then drawing at random a red ball is $1/4 \times 3/(3+1+6) = 0.25 \times 0.3 = 0.075$, and the probability of drawing the urn and then drawing at random, with replacement, first a red ball and then a blue ball is $1/4 \times 3/10 \times 1/10 = 0.25 \times 0.3 \times 0.1 = 0.0075$. Instead of urns containing coloured balls, the system uses documents containing terms, but the procedure is exactly the same.

4.1.7 An ideal user

The probability mechanisms that define how requests are generated from a relevant document should in some way reflect the way users choose the words when they formulate the request. When users enter a request in a full text information retrieval system, they do have a reasonable idea of what a relevant document would look like and they will choose the words accordingly (Ponte and Croft 1998). To formulate a request, users might picture themselves a relevant document to choose words from. A probability is assigned to each hypothesis "the user has the document in mind" and the documents are ranked by this probability (Miller et al. 1999). The document in the collection that is most similar to the document that the user has in mind is the best candidate for retrieval.

An *ideal* user might be defined as follows. Ideal users choose the relevant document they picture in their mind, and the corresponding query terms according to the probability mechanism that is informally introduced in the previous section. Ideal users know exactly what the collection looks like. Once they have decided which document they are looking for, they choose important terms and unimportant terms as defined above: the important terms are selected at random from the relevant document, and the unimportant terms are selected at random from the collection. Of course, ideal users do not exist in practice. Real users do not know what the collection looks like, and they often do not know

exactly what they are looking for. Thinking of the retrieval model as a model of an ideal user explains under which circumstances the model works best. According to the experimental results reported in chapter 5, 6 and 7, the ideal user assumption provides a reasonable approximation of the behaviour of the real world user. Similar reasons for using simplifications to real world problems exist in other research areas for very different problems. For instance in thermodynamics, an ideal gas consist of particals with zero volume that move in any direction with equal probability. Ideal gasses do not exist in practice, but in many cases they provide a convenient approximation of the real world: that is the essence of modelling.

4.1.8 An overview of this chapter

The remainder of this chapter is structured as follows. Section 4.2 formally introduces the basic model of the matching process. In section 4.3 translation of terms is added to model the query formulation process. Section 4.4 addresses the notion of importance of query terms. Section 4.5 and 4.6 will present the exact same models in terms of respectively hidden Markov models and Bayesian networks. Section 4.7 addresses implementation details and shows the resemblance of the resulting weighting formulas with other models. Finally, section 4.8 introduces extensions for proximity searching.

4.2 The basic retrieval model

This section formalises the basic retrieval model, that is, the model of the matching process. The section introduces respectively, the model's random variables and their sample spaces, the conditional independence assumptions and the specification of the probability measures.

4.2.1 Defining the probability space

Based on the informal description above, this section will define the basic probability measures that are used to rank the documents given a query. The model uses the following discrete random variables.

Definition 1 Let D be a discrete random variable "the document that the user has in mind", which sample space contains a finite number of points $\{d^{(1)}, d^{(2)}, \cdots, d^{(N)}\}$ each referring to an actual document in the collection.

Definition 2 Let I_i be a discrete random variable "importance of the ith query term", over the sample space $\{0, 1\}$, where 0 stands for unimportant and 1 for important.

Definition 3 Let T_i be a discrete random variable "the ith query term", which sample space contains a finite number of points $\{t^{(1)}, t^{(2)}, \cdots, t^{(m)}\}$ each referring to an actual term in the collection.

The notation $t^{(1)}$ is used to denote the actual first term in the system's vocabulary, for instance the term "aardvark" if the dictionary is sorted in alphabetical order. The notation t_1 is used to denote the realisation of the first term in the user's query, and changes per query.

At this point, one can argue that the retrieval model uses Luhn's similarity criterion (see section 2.3), because relevance information is not modelled explicitly. This is certainly true. In absence of relevance information, the similarity between query and document is the only information there is. If one of the documents in the collection is completely similar to the relevant document the ideal user has in mind, it is certainly relevant. Relevance information will be introduced more explicitly in section 4.4 in which sets of r relevant documents are modelled by using a separate random variable D_k ($1 \leq k \leq r$) for each relevant document.

4.2.2 Conditional independence assumptions

The joint probability $P(D, I_1, \cdots, I_n, T_1, \cdots, T_n)$ completely defines the information retrieval problem for a query of length n. According to the informal description of section 4.1, a query is generated by first selecting a document d with probability $P(D=d)$. Given that d is the document the user has in mind, tossing for importance and selecting the query terms is done independently for each query term i with respectively probability $P(I_i)$ and $P(T_i|I_i, D)$ as shown in equation 4.1.

$$P(D, \ I_1, \cdots, I_n, T_1, \cdots, T_n) = P(D) \prod_{i=1}^{n} P(I_i) P(T_i|I_i, D) \qquad (4.1)$$

The major difference between the language model-based retrieval model and the other retrieval models is that queries are explicitly modelled as sequences of query terms, not as sets of query terms. So, there is no special treatment of duplicate query terms, and relative positions of terms might in theory matter for the probability calculations. Of course, the relative positions do not matter in equation 4.1, because of the independence between query terms, but they do matter if for instance a simple bigram model is used as described in section 4.8.

Naive summing over all possible combinations of important and unimportant query terms would require 2^n additions, but fortunately sums can be distributed over the products as follows (McEliece and Aji 2000).

$$P(D, T_1, \cdots, T_n) = P(D) \prod_{i=1}^{n} \sum_{k=0}^{1} P(I_i=k) P(T_i|I_i=k, D) \qquad (4.2)$$

Ranking the documents by equation 4.2 will in fact rank the documents in decreasing order of the probability that the document is relevant given the query. This can be shown as follows by applying Bayes' rule. On the left-hand side, $P(D|T_1, T_2, \cdots, T_n)$ is the probability of D conditioned on the query

$T_1, T_2, \cdots, T_n$ of length n.

$$P(D|T_1, T_2, \cdots, T_n) = \frac{P(D)\ P(T_1, T_2, \cdots, T_n|D)}{P(T_1, T_2, \cdots, T_n)}$$

The denominator of the right-hand side of the formula does not depend on D. Therefore, documents might as well be ranked according to the numerator of the right-hand side, which is exactly what is done by equation 4.2. Note that in the traditional probabilistic model, the probability of relevance is defined by the probability of drawing a relevant document from a set of documents, for instance the set that is indexed with a certain term. The language model-based approach is not a set-based approach to retrieval. Therefore, it is only valid to talk about the probability of relevance of a specific document d.

4.2.3   Definition of the probability mechanism

The definition of the probability measures introduced above is quite straightforward. They are defined by using the number of documents in the collection and the term frequencies of a term in a document. The term frequency $tf(t,d)$ of a term t in a document d is defined as the number of times the term t occurs in the document d. Given the informal description of the probability mechanism presented in section 4.1, estimation of $P(D = d)$, $P(T_i = t_i|I_i = 1, D = d)$ and $P(T_i = t_i|I_i = 0)$ in equation 4.2 will be done as follows.

$$P(D = d) \quad = \quad \frac{1}{\#(\text{documents})} \tag{4.3}$$

$$P(T_i = t_i|I_i = 1, D = d) \quad = \quad \frac{tf(t_i, d)}{\sum_t tf(t, d)} \tag{4.4}$$

$$P(T_i = t_i|I_i = 0) \quad = \quad \frac{\sum_k tf(t_i, k)}{\sum_{t,k} tf(t, k)} \quad = \quad \frac{cf(t_i)}{\sum_t cf(t)} \tag{4.5}$$

As said in section 4.1, important terms are selected from the relevant document. The probability $P(T_i = t_i|I_i = 1, D = d)$ of selecting an important term is therefore defined by the number of occurrences of the term in the document divided by the length of the document. The probability of selecting an unimportant term does not depend on the relevant document. Unimportant terms are selected at random from the entire collection, so $P(T_i = t_i|I_i = 0)$ is defined by the number of occurrences $cf(t_i)$ of the term in the collection divided by the total length of the collection $\sum_t cf(t)$. In equation 4.5, $cf(t_i)$ is the collection frequency of the term t_i: the frequency of occurrence in the collection.[1]

4.2.4   Alternative definitions

Two alternatives to respectively equation 4.3 en 4.5 might be defined. The alternative to equation 4.3, the probability that a document is drawn at random

---

[1] In some publications (e.g. Sparck-Jones et al. 2000) the term 'collection frequencies' is also used to denote document frequencies.

from the collection, is based on the following observation. Suppose that the system has to find relevant documents to a query that only contains unimportant terms. In this case, the best thing the system probably can do is to give the user the longest documents. Long documents contain more information and therefore have a higher probability of containing information that is useful to the user.

$$P(D = d) \quad = \quad \frac{\sum_t tf(t,d)}{\sum_{t,k} tf(t,k)} \tag{4.6}$$

So, it is assumed that the marginal probability of a document being relevant $P(D{=}d)$ is proportional to its length. One might imagine the random selection of a document by the random selection of a term from the collection; whichever document contained the term is the selected relevant document.

The alternative of equation 4.5, the probability of drawing a term at random from the collection, is based on the following pragmatic observation. Most of the term weighting algorithms presented in section 2.4 use the document frequency $df(t)$ to include global information of terms. The document frequency $df(t)$ is defined by the number of documents in which the term t occurs.

$$P(T_i = t_i | I_i = 0) \quad = \quad \frac{df(t_i)}{\sum_t df(t)} \tag{4.7}$$

4.2.5 Unknown parameters

The probability that a term on position i in the query is important, $P(I_i{=}1)$, is not easily defined by a basic principle of the probability mechanism that is informally described in section 4.1. Therefore, the probabilities will be treated as the unknown parameters of the model, for which λ_i will be used.

$$P(I_i = 1) \quad = \quad \lambda_i \tag{4.8}$$

This also determines the probability of a term being unimportant as $P(I_i{=}0) = 1-\lambda_i$, but it does not explain how to determine the value of λ_i. The importance of a query term in a document is an event that cannot be observed directly. A query term is either important or unimportant, but there is no way that the system can know which query terms are the important terms and which query terms are the unimportant terms. For an ad-hoc query (when there are no previously retrieved documents to guide the search), the additional simplifying assumption is made that each query term i will be equally important, which leaves the model with only one unknown parameter λ. The exact value of λ will be determined empirically on some information retrieval test collection. If some relevant documents are known, the EM-algorithm presented in section 4.4 can be used to determine estimates λ_i for each query term. Some implications of this line of reasoning is further discussed in 4.4.

4.3   The extended retrieval model

This section adds translation of terms to the basic model presented in section 4.2. The concept of statistical translation of terms turns out to be a valuable tool to explain the use of structured queries.

4.3.1   Adding statistical translation

If the vocabulary of the request differs from the vocabulary of the document representations, an additional query formulation step has to account for the translation of the query terms to the request words. An extreme example of such a case is the situation where the user wants to do a cross-language search using French queries on an English database. To model this situation another random variable will be introduced.

Definition 4 Let S_i be a discrete random variable "the ith request word", which sample space contains a finite number of points $\{s^{(1)}, s^{(2)}, \cdots, s^{(m')}\}$ each one referring to an actual word in the vocabulary of requests.

The random variable S_i has as its sample space all words occurring in requests, whereas T_i has as its sample space all terms occurring in the queries and the document collection. The joint probability measure $P(D, I_1, \cdots, I_n, T_1, \cdots, T_n, S_1, \cdots, S_n)$ completely defines the information retrieval problem if a query of length n is entered. According to the informal description of section 4.1, a query is generated by first selecting a relevant document with probability $P(D)$. Given that D is the document the user has in mind, tossing for importance and selecting the terms is done independently for each term on position i with respectively probability $P(I_i)$ and $P(T_i|I_i, D)$. Given each T_i, selection of the request words S_i is assumed to be done conditionally independent from D and I_i given T_i with probability $P(S_i|T_i)$. This situation is formalised in equation 4.9.

$$P(D,\ I_1, \cdots, I_n, T_1, \cdots, T_n, S_1, \cdots, S_n)\ =$$
$$P(D)\prod_{i=1}^{n} P(I_i)P(T_i|I_i, D)P(S_i|T_i) \tag{4.9}$$

Summing over all possible translations and over all possible combinations of important and unimportant terms can again be done by distributing the sums over the products as follows, where m is the number of points in the sample space of T_i (Hiemstra and De Jong 1999).

$$P(D,\ S_1, \cdots, S_n)\ =$$
$$P(D)\prod_{i=1}^{n}\sum_{j=1}^{m} P(S_i|T_i\!=\!t^{(j)})\sum_{k=0}^{1} P(I_i\!=\!k)P(T_i\!=\!t^{(j)}|I_i\!=\!k, D) \tag{4.10}$$

A similar statistical translation model was introduced by Berger and Lafferty (1999). Their model differs from equation 4.10, because they smoothed (see section 4.4) the model with global information on S_i instead of global information on T_i.

4.3.2 Statistical translation in practice

In practice, the statistical translation model will be used as follows. The automatic query formulation process will translate the request $S_1, S_2, \cdots, S_n$ using a probabilistic dictionary. The probabilistic dictionary is a dictionary that lists pairs (s, t) together with their probability of occurrence, where s is from the sample space of S_i and t is from the sample space of T_i. For each S_i there will be one or more realisations t_i of T_i for which $P(S_i|T_i = t_i) > 0$, which will be called the possible translations of S_i. The possible translations should be grouped for each i to search the document collection, resulting in a structured query. For the example of cross-language information retrieval, suppose the original French request on an English collection is "déchets dangereux", then possible translations of "déchets" might be "waste", "litter" or "garbage", possible translations of "dangereux" might be "dangerous" or "hazardous" and the structured query can be presented as follows.

$$((\texttt{waste} \cup \texttt{litter} \cup \texttt{garbage}), \; (\texttt{dangerous} \cup \texttt{hazardous}))$$

The product from $i = 1$ to n (in this case $n = 2$) of equation 4.10 is represented above by using the comma as is done in the representation of a query of length 2 as T_1, T_2. The sum from $j = 1$ to m of equation 4.10 is represented by displaying only the realisations of T_i for which $P(S_i|T_i) > 0$ and by separating those by '$\cup$'. So, in practice, translation takes place during automatic query formulation, resulting in a structured query like the one displayed above that is matched against each document in the collection. Unless stated otherwise, whenever this chapter mentions 'query terms', it will denote realisations of T_i. Realisations of S_i, the 'request words', will usually be left implicit. The combination of the structured query representation and the translation probabilities will implicitly define the sequence of the request words $S_1, S_2, \cdots, S_n$, but the actual realisation of the sequence is not important to the system.

4.3.3 An extension of strict Boolean retrieval

The difference between the vocabularies of S_i and T_i is not an essential feature of the statistical translation model, but the fact that it allows for the modelling of structured queries *is* essential. The resemblance of the translated query and structured queries in the Boolean model is striking. If the Boolean model was to be used for a cross-language retrieval task, the obvious thing to do would be to build a faceted query that groups the possible translations of each term using the OR operator and conjoin the groups using the AND operator.

The statistical translation model defines a conjunction and disjunction operation that can be used to replace the Boolean operators as an extension of strict Boolean searching. Following this line of reasoning, unstructured queries of the simple model defined by equation 4.2 are assumed to be AND-queries. An OR-query consists of all realisations t_i of the random variable T_i in equation 4.10 for which $P(S_i|T_i = t_i) > 0$. The calculation of AND-queries uses the product of

probabilities, whereas the calculation of OR-queries uses the sum of probabilities.[2] If all words in the request are important,[3] that is, if $P(I_i\!=\!1) = 1$ for each request position i, then equation 4.10 will behave like the traditional Boolean model, assigning zero probability to documents that do not exactly match the structured query. The documents that do match the structured query will be assigned some probability higher than 0. All realisations of T_i that are not part of the structured query are assigned a translation probability of 0. For the realisations that are part of the query, the translation probabilities $P(S_i|T_i)$ might be any value higher than 0, indicating the probability that the term T_i actually generated the request word S_i.

4.3.4　On-line morphological expansion using a stemmer

Consider the example of a stemmed index and natural language request. During indexing the system might keep track of all pairs of stems and full forms to build a dictionary that translates full forms to stems. In this case, each word will have only one possible translation, so the resulting query in not structured in any way. However, the translation probability might improve the system.

More interestingly is the following weird example. Suppose that the user enters stems, and the index contains the full forms of the words from the documents. Again, during indexing the system might keep track of all pairs of stems and full forms to build a dictionary that translates stems to full forms. Note that the translation probabilities of all entries are 1, because each term in the collection generates one unique stem. By using the dictionary, the system can generate all possible morphological variants of each 'request stem' and group those for each i. For instance, the request (`funni`, `tabl`), which might be the result of stemming "funny tables" with the Porter stemmer, can be translated to form the following structured query.

$$((\texttt{funny} \cup \texttt{funnies} \cup \texttt{funniness}),\ (\texttt{table} \cup \texttt{tables} \cup \texttt{tabled}))$$

This will be called on-line stemming, or on-line morphological generation. As will be shown in section 4.7.3, this weird example of the user entering 'request stems' produces exactly the same information retrieval results as the traditional use of a stemmer during indexing and query formulation. Interestingly, since this thesis presents a model, may-be the only model, of the query formulation process, it might be concluded that stemmers have been used weirdly in information retrieval systems for the past 30 years.

4.3.5　Expansion with synonyms and related terms

As said before, often the actual realisation of the request words $S_1, S_2, \cdots, S_n$ will be left implicit to the system. Suppose however that there is in fact a

---

[2]Interestingly, when George Boole devised his system of logic, he called the AND and OR operators respectively the 'logical product' ($\times$) and the 'logical sum' ($+$) (see section 2.2).

[3]Note that, because of the one-to-one translation model an important query term generates an important request word, and an unimportant query term generates an unimportant request word.

large quantity of previously entered requests with corresponding relevance judgements. If such a corpus of documents and associated requests were available, the system could infer for instance that most documents that contain the terms `nuclear` and `energy` have corresponding requests that contain "atomic" and 'power". In fact, such corpora exist. They are called information retrieval test collections (see appendix A). Unfortunately, test collections only contain a very small number of requests compared to the number of documents, making the construction of a reliable probabilistic dictionary of synonyms and related terms problematic. Lacking such a corpus, Berger and Lafferty (1999) automatically generated synthetic training requests as random samples of a distribution that is based on some mutual information statistic. The resulting synthetic training data was used to construct a probabilistic translation dictionary that lists pairs of synonyms and related terms. For instance, the query "pope cuba" for Pope John Paul II's visit to Cuba in 1998 would be expanded by their system in the following structured query.

$$((\texttt{pope} \cup \texttt{pontiff} \cup \texttt{paul} \cup \texttt{john}), (\texttt{cuba} \cup \texttt{castro}))$$

4.3.6 Discussion

The extensions of the Boolean model presented in sections 2.3.4 and 2.3.5, the p-norm model and Paice's fuzzy set model, are quite different from the extension suggested in this section. Both models assume that unstructured queries are queries that are somewhere 'in between' AND-queries and OR-queries. The p-norm model and Paice's model can be reduced to the vector model by assigning a value of 1 to respectively both p parameters of p-norm, and both r parameters of Paice's model. Slightly higher values of these parameters will result in operators that are somewhat AND and somewhat OR. For a user it is hard to interpret an operator that is AND with e.g. p, or $r = 2$ as is done by some of the extensions of the Boolean model suggested in information retrieval literature.

The extension in this section suggests that AND-queries should rank a document by multiplying, and OR-queries by adding the probability of drawing terms from the document. The underlying probability mechanism is easy to understand and easy to explain. For instance, if a fair die is tossed twice, the probability of first a 5 *and* then a 6 should be calculated as $1/6 \times 1/6$. On each toss it is possible to specify more than one preferred outcome. For instance the probability of first a 4 *or* a 5, *and* then a 6 should be calculated as $(1/6 + 1/6) \times 1/6$. This line of reasoning needs the queries to be in conjunctive normal form, because the number of draws has to be unambiguously specified as well as which draw belongs to which query term. For instance the query (`a AND b`) `OR c` is not a valid query, because (`a AND b`) refers to drawing two terms from a document which contradicts with `OR c` which refers to drawing one term from a document. In fact, the reason that this query is invalid is closely related to the reason that for instance the query (`a AND b`) `ADJ c` is invalid in the traditional Boolean model (see section 3.3.2).

Automatic query formulation, for instance using a translation module or us-

ing a morphological component, will produce valid queries in conjunctive normal form by design. This might also be the case for manually formulated structured queries. For instance, the natural language query languages introduced in section 3.3.4 practically force the users to formulate their queries in conjunctive normal form. If the query language uses the traditional Boolean operators, manually formulated Boolean queries might be converted automatically to their conjunctive normal form. For the extended Boolean models presented in section 2.3, the p-norm model, Paice's fuzzy set model, and the inference network model all combinations of the standard Boolean operators AND and OR are valid. However, for these models, the distributive laws that hold for conventional Boolean expressions are not valid. For instance, `(a AND c) OR c` and `(a OR c) AND (b OR c)` are equivalent in the traditional Boolean model, but this is not generally the case for the extended Boolean models presented in section 2.3.

Section 4.7 will show the following. For each indexing strategy that unambiguously converts words in documents to index terms, there is a corresponding query formulation strategy that produces the exact same results on an index that did *not* use this indexing strategy. Unambiguously in this context means that each word will be converted deterministically to one, and only one term e.g. as done by converting words to lower case or as done by a stemmer. For instance, the weird on-line morphological generation as described above should produce exactly the same retrieval results as off-line stemming. This is the case for strict Boolean searching, but this is not generally the case with the extended Boolean models presented in section 2.3. Section 4.7 will show that the language modelling extension will produce the exact same probabilities and therefore the exact same ranking for on-line generation and off-line stemming. This implies that for instance a wildcard search for `dog∗` will produce the exact same results as a hypothetical indexing and automatic query formulation process that converts each word beginning with the characters `d`, `o`, `g` to the term `dog`.

4.3.7   Extension of the Boolean NOT

The metaphor of drawing terms at random from documents provides a natural extension of the Boolean NOT. For instance a search for `development NOT sustainable` might be modelled by a probability mechanism in which first the term `development` is drawn at random from the document and then any term except for the term `sustainable`. The probability that the next query term is not t_i is calculated as follows.

$$
\begin{aligned}
P(T_i \neq t_i | I_i = 1, D = d) &= 1 - P(T_i = t_i | I_i = 1, D = d) \\
P(T_i \neq t_i | I_i = 0) &= 1 - P(T_i = t_i | I_i = 0)
\end{aligned}
$$

Like the extension of the Boolean OR, the NOT operator should always refer to the position in the query to which it applies. So, `a NOT (b AND c)` is not a valid query, whereas `a NOT (b OR c)` is a valid query. One might argue that the extension of the NOT operator is modelled by a disjunction of all terms, except for terms specified within the not.

The extension of AND and OR behaves like the traditional strict Boolean model if all terms are important terms. This is not the case for the NOT operator. Even if the draw of any term except for the term `sustainable` in the example above is an important term with $P(I_i = 1) = 1$, then this will still match every document in the collection because every document will contain some terms that are not the term `sustainable`. If users want a strict Boolean NOT, the system should provide two separate operators, one with the strict interpretation and one as specified above. For full-text retrieval however, queries using NOT are generally quite rare. the manually formulated Boolean queries used for the experiments in section 5 only have one occurrence of the NOT operator in 50 queries.

4.4 Importance of query terms

The importance of query terms is one of the key-concepts of the language modelling approach presented in this book. It can be used to explain mandatory terms, stopping, coordination level ranking of short queries and relevance feedback. First, some simplified notations will be introduced.

4.4.1 Simplified notations

The previous sections introduced a rather elaborate notation to describe the new retrieval model. The notation differs considerably from the notations used in earlier publications (Hiemstra 1998a) and from the notation used in the publications by Miller, Leek, and Schwartz (1999), Song and Croft (1999) and Berger and Lafferty (1999). The old notations will be reintroduced at the end of this paragraph. The reason for presenting a new notation is two-fold. Firstly, the notation used in this chapter is more explicit in the assumptions made and therefore mathematically more precise. Secondly, the notation explicitly introduces a new concept in retrieval modelling: the *importance* of a query term. However, the notation that is used to introduce the importance of a query term is not very readable. Therefore, the following notations will be used as a short-hand.

$$
\begin{aligned}
\lambda_i &\quad \text{instead of} \quad P(I_i = 1) \\
1 - \lambda_i &\quad \text{instead of} \quad P(I_i = 0) \\
P(T_i|D) &\quad \text{instead of} \quad P(T_i|I_i = 1, D) \\
P(T_i) &\quad \text{instead of} \quad P(T_i|I_i = 0)
\end{aligned}
\tag{4.11}
$$

The simplified notations of the probability measures are more in correspondence with the way the probabilities are actually defined than the elaborate notation. The simplified notation is intuitively easy to understand. It will be used throughout the rest of this book. Substituting the probability measures of equation 4.2 by their simplified versions results in the following definition of

the matching model.

$$P(D, T_1, T_2, \cdots, T_n) \;=\; P(D) \prod_{i=1}^{n} ((1-\lambda_i)P(T_i) + \lambda_i P(T_i|D)) \qquad (4.12)$$

Similarly, the simplified notation of the statistical translation information retrieval model of equation 4.10 results in the following definition of the extended model.

$$P(D, S_1, S_2, \cdots, S_n) \;= \\ P(D) \prod_{i=1}^{n} \sum_{j=1}^{m} P(S_i|T_i \!=\! t^{(j)})((1-\lambda_i)P(T_i \!=\! t^{(j)}) + \lambda_i P(T_i \!=\! t^{(j)}|D)) \quad (4.13)$$

In publications of Hiemstra (1998a), Song and Croft (1999) and Berger and Lafferty (1999), the parameter λ is called a smoothing parameter. Smoothing parameters are used in probability estimators to move the estimates away from maximum likelihood estimates (Manning and Schütze 1999). This book presents the parameter λ as the probability of term importance and argues that all definitions should use maximum likelihood estimates as done by Miller et al. (1999). Of course, this does not really make the model very different, but it does make the interpretation of the model different and it might lead to a better understanding of information retrieval. One practical difference from all of the publications mentioned in this paragraph, is that each query term i might be assigned a different λ_i, of which the value can be determined from some examples of relevant documents.

4.4.2   Relevance weighting

Documents that are judged relevant by the user can be used to re-estimate the importance weights for each i separately. Since the importance of terms given a document is an event that cannot be observed directly, it is necessary to resort to methods for the estimation of probabilities from incomplete data. A standard method for finding maximum likelihood estimates from incomplete data is the Expectation Maximisation (EM) algorithm (Dempster, Laird, and Rubin 1977). The general idea of the EM-algorithm is that if only the expected values of the unobserved data were known, then these values could be used to estimate the probabilities we want to know. Unfortunately, in order to compute the expected values the probabilities are needed. To break the vicious circle, the EM-algorithm takes any set of probabilities to compute the expected values; these are used to re-estimate the probabilities. The new probabilities are used to compute new expected values, etc. The two steps, called expectation step and maximisation step are repeated until the probabilities do not change significantly anymore. The algorithm is guaranteed to converge to a local maximum.

The EM-algorithm will be applied as follows. The algorithm should maximise for each document the probability that the user has the document in mind when he/she entered the query. Strictly speaking, the model as presented up till

now does not allow that the user has more than one document in mind, because the documents $d^{(1)}, d^{(2)}, \cdots$ of the sample space of D are mutually exclusive. Therefore r separate random variables D_j, $(1 \leq j \leq r)$ will be introduced, one for each relevant document.

Definition 5 Let D_j be a discrete random variable "the jth relevant document", which sample space contains a finite number of points $\{d^{(1)}, d^{(2)}, \cdots, d^{(N)}\}$ each referring to an actual document in the collection.

Definition 6 Let I_{ij} be a discrete random variable "the importance of the ith query term in the jth relevant document", over the sample space $\{0, 1\}$, where 0 stands for unimportant and 1 for important.

Dempster et al. (1977) describe the algorithm by defining the observed data as the 'incomplete data' and the combination of the observed data and the unobserved data as the 'complete data'. The incomplete data consist in this case of a sequence of n query terms T_i, $(1 \leq i \leq n)$, and a set of r relevant documents D_j, $(1 \leq j \leq r)$. The complete data 'completely' define how the model generated the observed data. It consists of the query terms, the relevant documents and r sequences of n binary random variables I_{ij}, $(1 \leq i \leq n; 1 \leq j \leq r)$ indicating the importance of each term on position i in the relevant document j. The values $\lambda_1, \cdots, \lambda_n$ that the algorithm tries to find are related to the complete data by the following likelihood function, assuming independence between the observed relevant documents:

$$
\prod_{j=1}^{r} P(D_j = d_j, T_1 = t_1, \cdots, T_n = t_n, I_{1j} = x_{1j}, \cdots, I_{nj} = x_{nj})
$$

$$
= \prod_{j=1}^{r} P(D_j = d_j) \prod_{i=1}^{n} \Big(((1 - \lambda_i) P(T_i = t_i))^{(1 - x_{ij})} (\lambda_i P(T_i = t_i | D_j = d_j))^{x_{ij}} \Big)
$$

$$
= \Big(\prod_{j=1}^{r} P(D_j = d_j) \Big) \Big(\prod_{i=1}^{n} \prod_{j=1}^{r} P(T_i = t_i | D_j = d_j) P(T_i = t_i) \Big)
$$

$$
\Big(\prod_{i=1}^{n} \lambda_i^{\sum_{j=1}^{r} x_{ij}} \Big) \Big(\prod_{i=1}^{n} (1 - \lambda_i)^{r - \sum_{j=1}^{r} x_{ij}} \Big)
$$

The complete-data sufficient statistics are n counts M_i for which $M_i = \sum_{j=1}^{r} I_{ij}$ $(1 \leq i \leq n)$, and the unknown parameters λ_i can be estimated from the realisation of the complete-data sufficient statistics by: $\lambda_i = m_i / r$. This will be the maximisation step. The expectation step will estimate the complete-data sufficient statistics, by the following expectation.

$$
E(M_1, \cdots, M_n | T_1, \cdots, T_n, D_1, \cdots, D_r, \lambda_1, \cdots, \lambda_n)
$$

Since the expectation of a sum is the sum of an expectation, the expectation of the importance of terms in each separate relevant document might be summed instead.

$$= \sum_{j=1}^{r} E(I_{1j}, \cdots, I_{nj} | T_1, \cdots, T_n, D_1, \cdots, D_r, \lambda_1, \cdots, \lambda_n)$$

Because of independence between terms, it is not necessary to enumerate all 2^n combinations of important and unimportant terms. Instead the expected values can be computed for each term independently. The resulting algorithm is displayed in figure 4.2. The algorithm iteratively maximises the probability of the query $t_1, t_2, \cdots, t_n$ given r relevant documents $d_1, d_2, \cdots, d_r$. Before the iteration process starts, the importance weights are initialised to their default values $\lambda_i^{(0)}$, where i is the position in the query. Each iteration p estimates a new relevance weight $\lambda_i^{(p+1)}$ by first doing the E-step and then the M-step until the value of the relevance weight does not change significantly anymore.

$$
\begin{array}{ll}
\text{E-step:} & m_i = \sum_{j=1}^{r} \dfrac{\lambda_i^{(p)} \cdot P(T_i = t_i | D_j = d_j)}{(1 - \lambda_i^{(p)}) P(T_i = t_i) + \lambda_i^{(p)} P(T_i = t_i | D_j = d_j)} \\[4ex]
\text{M-step:} & \lambda_i^{(p+1)} = \dfrac{m_i}{r}
\end{array}
$$

Figure 4.2: Relevance weighting for the basic model: EM-algorithm

A similar relevance weighting algorithm can be developed for structured queries of the statistical translation retrieval model by following the procedure as above. In this case, both the translations and the importance of the terms cannot be observed directly. The EM-algorithm estimates the translation probabilities $\tau_i(j)$ of jth possible translation $t^{(j)}$ of the request word on position i, and the probabilities λ_i of importance of the request word on position i. The algorithm displayed in table 4.3 iteratively maximises the model for query $t_1, t_2, \cdots, t_n$ of length n and r relevant documents $d_1, d_2, \cdots, d_r$. Before the iteration process starts, the importance weights and the translation probabilities are initialised to their default values $\lambda_i^{(0)}$ and $\tau_i(j)^{(0)}$, where i is the position in the query and j is the jth translation. Each iteration p estimates a new importance weight $\lambda_i^{(p+1)}$ by first doing the E-step and then the M-step until the values do not change significantly anymore. Translation probabilities that are initialised to zero, that is terms that are not in the structured query, will remain zero during and after reestimation. The same thing goes for the importance weights for that matter. Importance weights that are initialised to zero, that is request words that are treated as stop words, will remain zero even if the relevant documents contain many occurrences of the possible translations of the word.

The application of the EM-algorithm to the estimation of unknown parameters is standard practice in many applications of statistical language models. Often, a less broadly applicable version is used, for instance the Baum-Welch algorithm that was developed for the estimation of hidden Markov model pa-

$$m_i = \sum_{k=1}^{r} \frac{\lambda_i^{(p)} \left(\sum_{l=1}^{m} \tau_i(l)^{(p)} P(T_i = t^{(l)}|D_k = d_k) \right)}{\sum_{l=1}^{m} \tau_i(l)^{(p)} \left((1-\lambda_i^{(p)}) P(T_i = t^{(l)}) + \lambda_i^{(p)} P(T_i = t^{(l)}|D_k = d_k) \right)}$$

E-step:

$$n_i(j) = \sum_{k=1}^{r} \frac{\tau_i(j)^{(p)} \left((1-\lambda_i^{(p)}) P(T_i = t^{(j)}) + \lambda_i^{(p)} P(T_i = t^{(j)}|D_k = d_k) \right)}{\sum_{l=1}^{m} \tau_i(l)^{(p)} \left((1-\lambda_i^{(p)}) P(T_i = t^{(l)}) + \lambda_i^{(p)} P(T_i = t^{(l)}|D_k = d_k) \right)}$$

$$\lambda_i^{(p+1)} = \frac{m_i}{r}$$

M-step:

$$\tau_i(j)^{(p+1)} = \frac{n_i(j)}{r}$$

Figure 4.3: Relevance weighting for the extended model: EM-algorithm

rameters (Manning and Schütze 1999). Experimental results of the algorithm for the basic model (see section 5) indicate that in a few cases, the model degrades performance even if training data and test data are the same. Similar problems have been noted with language models for part-of-speech tagging (Elworthy 1994). The problem might be related to the the maximum likelihood criterion that underlies the EM-algorithm (Jelinek 1997, page 72). The maximum likelihood criterion is not directly related to the aim of maximising the probability of relevance and so it might not lead to it. A useful alternative criterion might be the maximum mutual information criterion which is successfully applied to speech recognition (Rabiner 1990). Instead of maximising the observed data, the criterion tries to minimise the model's average uncertainty of what the relevant document is. The criterion might result in better results but is hard to apply because it needs information from relevant and nonrelevant documents. In practice however, the basic EM-algorithm is as effective as relevance weighting for the traditional probabilistic model, which seems to make similar mistakes in a retrospective relevance weighting experiment.

4.4.3 Ranging from exact matching to stopping

Instead of providing some examples of relevant documents, the user might directly assign a value to λ_i. The concept of important and unimportant query terms is intuitively easy to understand for naive users of retrieval systems. Assigning an importance weight of 1 to a term will have the same effect as the exact match operator presented in section 3.3.4. The document should contain the term. Documents that do not contain the term are assigned zero probability and are therefore not retrieved. Assigning an importance weight of 0 to a term will be like treating the term as a stop word presented as in section 3.2.2. The term will have no effect on the final ranking.

Note that the model completely separates the importance of query terms from their frequency in the collection. In principle, the importance of a term does not necessarily have anything to do with the frequency of occurrence in the collection. For retrieval experts this needs getting used to, but for naive users this is an important advantage. There is something to the importance of query terms that has nothing to do with its frequency of occurrence in the collection. This 'something' is embodied in the value of λ_i. Sometimes a term, whether frequent or infrequent, simply does not occur in any of the relevant documents. The importance of a term *is* directly related to the term's distribution over relevant and non-relevant documents. Some words, like "the" or "and" are generally unimportant words, not because their collection frequency is high, but because their distribution in relevant documents is similar to their distribution in non-relevant documents. The same goes for some infrequent words like e.g. the word "presumably". As long as the relevant documents are unknown, all query terms, whether frequent or infrequent, might be assumed equally important. The retrieval model makes sure that the impact of each term on the final ranking will be based on their frequency of occurrence in the collection. Alternatively, some words like e.g. "the", "and" and "presumably" might be assigned an importance weight of 0. These words will not affect the ranking and therefore might as well be removed from the query as is done with stop words.

4.4.4   Coordination level ranking

If no information on relevant documents is available, the importance weights should be constant for each position i in the query that does not contain a stop word. The optimum value of the constant, λ, might change for different applications. A high value of λ results in rankings that obey the conditions of coordination level ranking (Hiemstra 2000). Coordination level ranking partially ranks documents in such a way that documents containing n query terms are always ranked above documents containing $n-1$ query terms. For low relevance weights however, chances are that documents that contain $n-1$ query terms are ranked above documents that contain n query terms. According to studies of user preferences, users do not like systems that do not obey the conditions of co-ordination level ranking. These problems become particularly apparent if short queries are used (Rose and Stevens 1997). In a lot of practical situations short queries are the rule rather than the exception, especially in situations where there is no or little user training like with Web-based search engines. High relevance weights might therefore be a good choice for applications in which very short queries are used, like web search engines. For some research groups, the interest of users in coordination level is the reason for developing ranking methods that are based on the lexical distance of search terms in documents instead of on frequencies of terms (Hawking and Thistlewaite 1996; Clarke et al. 1997). As pointed out by experiments of Wilkinson et al. (1996), some $tf.idf$ measures behave more like coordination level ranking than others. For instance, the Okapi BM25 algorithm behaves more like coordination level ranking than

the Smart tfc.nfc algorithm (see section 2.4). They showed that weighting measures that are more like coordination level ranking perform better on the TREC collection, especially if short queries are used. Following the results of Wilkinson et al. (1996) it might be useful to investigate what exactly makes a weighting measure "like" coordination level ranking. Appendix B sketches a proof that a high value of λ guarantees coordination level ranking.

4.4.5 Relation to previous work

Importance weighting as presented above is closely related to relevance weighting of the traditional probabilistic model presented in section 2.3.3. Both approaches try to use the distribution of terms over relevant and non-relevant documents to estimate term weights. The Robertson/Sparck-Jones relevance weight is different from the importance weight as it might range from minus infinity to infinity, but one of its components is quite similar: the probability of term occurrence given relevance. A simple approach to importance weighting might assume that a term is important if it occurs in the relevant document and unimportant otherwise. If so, the importance weight λ_i of the language models is equal to $P(D_k|L) = p_i$, the probability of term occurrence given relevance of the traditional probabilistic model, where R is the number of relevant documents and r_i is the number of relevant documents containing the term.

$$\lambda_i \;=\; p_i \;=\; \frac{r_i}{R} \tag{4.14}$$

As said in section 4.2.3, in absence of relevance information it will be assumed that the value of λ_i is fixed for each position i in the query. Assuming a fixed value for the traditional probabilistic model's p_i in absence of relevance information was suggested by Croft and Harper (1979). Often a constant value of $p_i = 0.5$ is used for the traditional probabilistic model if no relevance information is available. This results in negative weights for terms that occur in more than half of the documents in the collection. A lower constant might be more realistic, but also results in more terms with negative contributions, which might not be desirable for any term in absence of relevance information (Robertson and Walker 1997). Terms cannot have a negative contribution in the language model-based system.

The relation between importance weighting and relevance weighting is even stronger in the extension of the traditional probabilistic model presented in section 2.3.7. This model uses the binary event eliteness suggested by Harter's 2-Poisson model (see section 2.3.6). Given relevance, the probability of term eliteness is used like the probability of term importance, as the source of a mixture model. As shown in equation 2.15, the within document term frequencies are modelled by a mixture of two Poisson distributions, similar to the mixture model presented in this chapter. Term eliteness and term importance are both unknown, unobservable events, on which respectively the production of documents and the production of queries depend. However, they do not refer to the same event. If a query term does not occur in a relevant document, then it has

to be an unimportant term, but this term might still be elite if the document treats the subject referred to by the term to some extent.

4.5   Presentation as a hidden Markov model

In the last two decades, hidden Markov models have been successfully applied to numerous natural language processing tasks, like e.g. speech recognition, part-of-speech tagging and optical character recognition. This section briefly presents the retrieval model in terms of hidden Markov models as done by Miller et al. (1999). Hidden Markov models are often presented, and graphically displayed, as probabilistic finite state machines. Each state transition is assigned a probability. The state transitions generate an output sequence by some probability function as well. The models are called *hidden* Markov models, because only the output symbols can be observed, but not the underlying state sequence. For information retrieval, the output symbols are the query terms which are assumed to be produced by some unknown state sequence. This section is largely based on a tutorial by Rabiner (1990).

4.5.1   The basics

In general, an N-state hidden Markov model with M possible output symbols is described by three probability measures A, B and Π. The probability measure A has as its parameters the transitions probabilities a_{ij} $(1 \leq i, j \leq N)$. The probability measure B has as its parameters the observation probabilities $b_j(k)$ $(1 \leq j \leq N; 1 \leq k \leq M)$. The probability measure Π has as its parameters the initial state probabilities π_i $(1 \leq i \leq N)$.
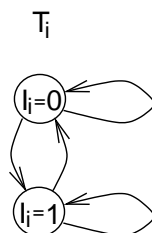
Figure 4.4: Document model as a two-state hidden Markov model

For information retrieval, each document is assigned a separate hidden Markov model. The simplest document model can be interpreted as the two-state hidden Markov model shown above if λ_i is constant $(1 \leq i \leq n)$ (Miller et al. 1999). The model has one state for the unimportant terms and one state for the important terms. The observation probabilities are estimated as $P(T_i|I_i = 0)$ for the unimportant terms and $P(T_i|I_i = 1, D = d)$ for the important terms, presented in section 4.2.3. The two transition probabilities to the state of the

important terms are estimated as $P(I_i=1) = \lambda$ and the two transition probabilities to the state of the unimportant terms are estimated as $P(I_i=0) = 1-\lambda$, as presented in section 4.2.3. States are graphically displayed as nodes of a graph, and state transitions with a probability higher than zero are displayed by an arc from one node to the other as shown in figure 4.4. The model has two different state transition values defined by only one parameter λ.

Tied state transitions

Note that generally a two-state hidden Markov model may have four different state transition values, defined by two parameters a_{11} (the probability of going from state 1 to state 1) and a_{22}, and the fact that $a_{12}+a_{11} = 1$ and $a_{22}+a_{21} = 1$. In the hidden Markov model displayed above, a_{12} is constrained to be equal to a_{22}. The state transitions are said to be *tied*. By tying the state transitions, the probability of going to a state does not depend on the (previous) state the model is in. This makes the model memoriless: a degenerate case of a hidden Markov model. Note that state transitions are also tied across all document models.

Fixed parameters

The number of free parameters of a hidden Markov model can be reduced by fixing the values of some parameters, so that the values may not be changed during training. For the retrieval model all output parameters (the probability measure B) will be fixed, because they describe the physical characteristics of the document collection which should only change if the collection, or a document in the collection, changes.

4.5.2 Left-right models

Each output symbol can be given its own states by expanding the two-state model of figure 4.4 to the $2n$-state model of figure 4.5. The resulting model is called a left-right model. Suppose that the states in figure 4.5 are somehow numbered from left to right, then state transition probabilities a_{ij} from a state i to a state j for which $j < i$ will be zero. In fact, the model displayed in figure 4.5 is a strict left-right model as $a_{ij} = 0$ for $i = j$ as well. Again, state transitions to the same state are tied, making the model a memoriless process. The random variables of the output symbols are displayed above the corresponding state transitions.

For the statistical translation model, the output symbols are modelled by the stochastic variable S_i. Each S_i may have several possible translations that should be assigned hidden states as well. There should be a state for each possible term in the vocabulary of the documents. The resulting left-right model of the statistical translation information retrieval model is graphically displayed in figure 4.6. Again, random variables of the output symbols are displayed above the corresponding state transitions. Transitions to the model's $I_i=0$ and $I_i=1$
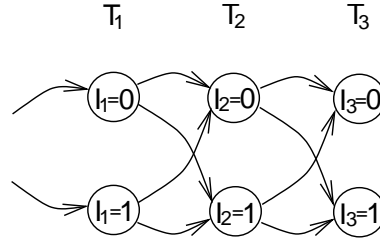
Figure 4.5: Document model as a left-right hidden Markov model

states do not produce an output symbol. These transitions are called epsilon transitions.
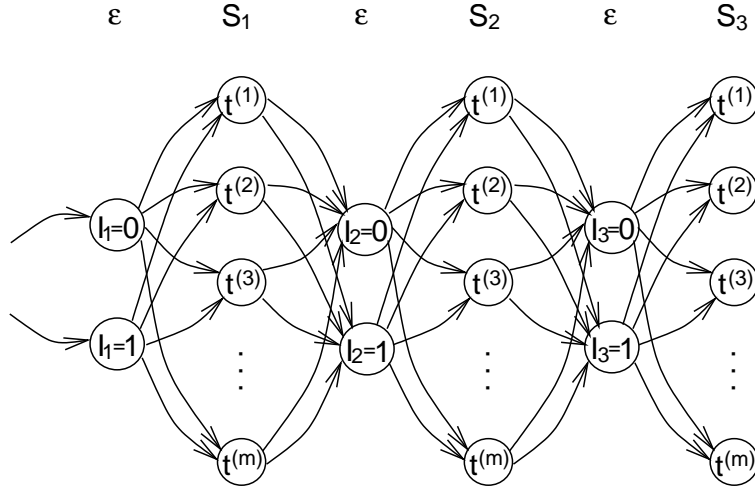
Figure 4.6: Extended document model as a hidden Markov model

4.5.3   Application of hidden Markov model theory

There are three basic problems of interest to be solved for a hidden Markov model in order to be useful in real-world applications. These problems are the following (Rabiner 1990; Manning and Schütze 1999).

1. Given the observation sequence $s_1, s_2, \cdots, s_n$ and a document model, how to efficiently compute the probability of the observation sequence given the model?

2. Given the observation sequence $s_1, s_2, \cdots, s_n$ and a document model, how to choose a corresponding state sequence that best explains the model?

3. How to adjust the parameters (A, B, Π) of the document model to maximise the probability of a given observation sequence $s_1, s_2, \cdots, s_n$?

Problem 1 is the search problem in information retrieval. It involves the distribution of sums over products as shown by equation 4.2 which is usually called the forward algorithm (or alternatively, the backward algorithm). Problem 2 is not so interesting for the information retrieval models. It decides for each term its importance and its most probable translation in the document. It usually involves the Viterbi algorithm. The third problem is the relevance feedback problem in information retrieval. It describes the case in which some documents are known to be relevant, and their parameters have to be optimised for the given output sequence. It is used to re-estimate the importance weight of each query term. This involves the EM-algorithm presented in section 4.4.2, which is a general version of the Baum-Welch algorithm that is specifically designed for hidden Markov models (Rabiner 1990).

4.5.4 Discussion

Hopefully, this section was an eye opener for readers that gained experience with hidden Markov models on other applications. Remember that this section presented the exact same model from a different perspective. A number of little tweaks to hidden Markov model theory were needed to describe the model correctly: tied state transitions, fixed parameters and epsilon transitions. These tweaks are also extensively used in speech recognition (Rabiner 1990). It was also noted that the models as presented above are memoriless systems and therefore degenerate cases of hidden Markov models.

4.6 Presentation as a Bayesian network

Bayesian networks were successfully applied in the mid 1980's and the 1990's to a wide variety of applications, ranging from for instance medical expert systems (Heckerman 1991) to error correction codes (McEliece, MacKay, and Cheng 1998). This section briefly presents the retrieval model in terms of Bayesian networks. Previous work on the application of Bayesian networks to information retrieval was presented in section 2.3.8.

4.6.1 The basics

A Bayesian network is an acyclic directed graph that encodes probabilistic dependency relationships between random variables. A directed graph is acyclic if there is no directed path $A \rightarrow \cdots \rightarrow Z$ such that $A = Z$. The presentation of probability distributions as directed graphs, makes it possible to analyse complex conditional independence assumptions by following a graph theoretic approach. Probability theory ensures that the system as a whole is consistent. Some alternative names for Bayesian networks are belief networks, probabilistic independence networks, influence diagrams and causal nets (Pearl 1988).

Figure 4.7 graphically displays the basic retrieval model of section 4.2 as a Bayesian network. It should not be confused with the graphical representation
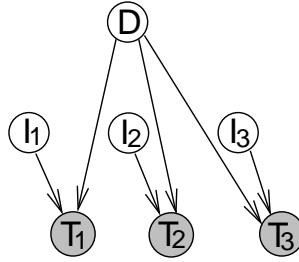
Figure 4.7: Retrieval model as a Bayesian network

of the hidden Markov model introduced earlier, in which each node represents
the value of a state variable and arcs represent state transitions. In figure 4.7,
the nodes represent random variables and arcs represent dependence relations.
Following the convention used by Jordan (1998), clear nodes are used to rep-
resent unknown, hidden variables and shaded nodes are used to denote known,
observed variables.
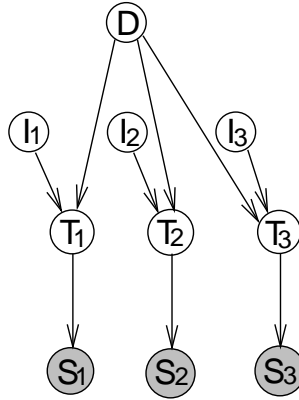
Figure 4.8: Retrieval model as a Bayesian network, including translation

Figure 4.8 displays the extended retrieval model. The graphical representa-
tion shows that, for each i, the terms T_i are conditionally independent given D
and I_i. The translation to each word S_i is in turn conditionally independent
from D and I_i given T_i.

4.6.2   Discussion

Remember again that this section presented the exact same retrieval models
from a different perspective. The application of Bayesian networks to infor-
mation retrieval was discussed before in section 2.3.8. The model presented
above shares some important features with previously introduced Bayesian net-
works for information retrieval. For instance, the model infers the probability

of the query from the hypothesis that a document is relevant as is done by Turtle (1991). The model differs considerably from previous work for two main reasons.

Firstly, it does not imply a commitment to the Bayesian approach to probability and statistics. The success of the theory presented in this book, should in the author's opinion mainly be contributed to the 'dumb' formulation of an explicit probability mechanism in section 4.1 that accounts for the estimation of classical probabilities in an intuitively plausible way.

The second difference with previous publications is that the models of figure 4.7 and figure 4.8 are tractable. Estimates of the probability of relevance can be computed rather trivially in linear time by distributing sums over products as is done in the belief propagation algorithm introduced by Pearl (1988) for complex Bayesian networks. It is therefore unnecessary to introduce approximate link matrices as e.g. done by Turtle (1991) for the Inquery model.

4.7 From probability measure to term weighting algorithm

Similar to the traditional probabilistic model presented in section 2.3.3, the probability measures for ranking documents can be rewritten into a format that is easy to implement. A presence weighting scheme (Robertson and Sparck-Jones 1976) (as opposed to a presence/absence weighting scheme) assigns a zero weight to terms that are not present in a document. Presence weighting schemes can be implemented using the vector product formula. This section presents the resulting algorithms.

4.7.1 Relation to *tf.idf* and relevance weighting

First, let's have a look again at the simplified notation of the basic probability measure as introduced by equation 4.12:

$$P(D, T_1, T_2, \cdots, T_n) \;=\; P(D) \prod_{i=1}^{n} ((1-\lambda_i)P(T_i) + \lambda_i P(T_i|D))$$

Dividing the formula by $\prod_{i=1}^{n}((1-\lambda_i)P(T_i))$ will not affect the ranking because λ_i and $P(T_i)$ have the same value for each document. Doing so results in a document ranking function that is somewhat similar to Ng's likelihood ratio formula (Ng 2000).

$$P(D, T_1, T_2, \cdots, T_n) \;\propto\; P(D) \prod_{i=1}^{n} (1 + \frac{\lambda_i P(T_i|D)}{(1-\lambda_i)P(T_i)})$$

Any monotonic transformation of the document ranking function will produce the same ranking of the documents. Instead of using the product of weights, the formula can be implemented by using the sum of logarithmic weights. Doing so

and replacing $P(D)$, $P(T_i|D)$ and $P(T_i)$ by the definitions in equations 4.3, 4.4 and 4.5 results in:

$$P(D{=}d, T_1{=}t_1, T_2{=}t_2, \cdots, T_n{=}t_n) \propto \sum_{i=1}^{n} \log(1 + \frac{\lambda_i tf(t_i, d) \sum_t cf(t)}{(1{-}\lambda_i)cf(t_i) \sum_t tf(t, d)}) \qquad (4.15)$$

The formula above will assign zero weight to each term t_i for which $tf(t_i, d) = 0$, since $\log(1) = 0$. In the formula, the definition of $P(D)$ can be ignored, because it is constant for any document d. If the alternative equation 4.6 of section 4.2.4 is used, then $P(D)$ can no longer be ignored, except for its denominator $\sum_{t,k} tf(t, k)$. The resulting presence formula will rank the entire collection, only assigning a zero score to documents of length 1, whose sole term is not among the query terms.

$$P(D{=}d, T_1{=}t_1, T_2{=}t_2, \cdots, T_n{=}t_n) \propto$$
$$\sum_t tf(t, d) + \sum_{i=1}^{n} \log(1 + \frac{\lambda_i tf(t_i, d) \sum_t cf(t)}{(1{-}\lambda_i)cf(t_i) \sum_t tf(t, d)})$$

Equation 4.7 of section 4.2.4 results in an algorithm that uses document frequencies instead of collection frequencies. Document frequencies are familiar from other retrieval models, and are used in most of the existing term weighting algorithms. The resulting presence weighting algorithm is the following.

$$P(D{=}d, T_1{=}t_1, T_2{=}t_2, \cdots, T_n{=}t_n) \propto \sum_{i=1}^{n} \log(1 + \frac{\lambda_i tf(t_i, d) \sum_t df(t)}{(1{-}\lambda_i)df(t_i) \sum_t tf(t, d)})$$

The formula can be interpreted as a *tf.idf* term weighting algorithm with document length normalisation. Also, the formula can be interpreted as using the odds of the probability of term importance which, as said in section 4.4.5, might be approximated by the probability of term occurrence given relevance of the traditional probabilistic model. Using the vocabulary of the vector space model and the traditional probabilistic model, the weighting function might be interpreted as follows:

$\dfrac{tf(t_i, d)}{df(t_i)}$ is the *tf.idf* weight of the term t_i in the document d

$\dfrac{1}{\sum_t tf(t, d)}$ is the inverse length of document d

$\dfrac{\lambda_i}{1 - \lambda_i}$ is the odds of the probability of term importance given relevance

$\sum_t df(t)$ is constant for any document k and term t_i. It is calculated once for the collection.

The query weights of the vector product formula can be used to account for multiple occurrences of the same term in the query. The resulting vector product version of the ranking formula is displayed in figure 4.9 in a similar informal way as done in section 2.4 for other term weighting algorithms.

$$
\begin{aligned}
\text{vector product:} \quad & \text{score}(\vec{d}, \vec{q}) \;=\; \sum_{k=1}^{m} d_k \cdot q_k \\[2mm]
\text{query term weight:} \quad & q_k = tf \\[2mm]
\text{document term weight:} \quad & d_k = \log(1 + \frac{tf \;\cdot\; (\text{sum of } df\text{'s})}{df \cdot \text{document length}} \cdot \frac{\lambda_k}{1 - \lambda_k})
\end{aligned}
$$

Figure 4.9: *tf.idf*-like term weighting algorithm

4.7.2 Discussion

The purpose of this section is *not* to show that the language modelling approach to information retrieval is so flexible that it can be used to model or implement many other approaches to information retrieval. For this reason, it differs considerably from other publications that also compare retrieval models within one framework (Turtle and Croft 1992; Wong and Yao 1995). Although this section claims that the language modelling approach may result in *tf.idf* term weighting, the *tf* component and the *idf* component both fall within the logarithm, making it a *tf + idf* algorithm rather than a *tf.idf* algorithm. Also, as shown in section 4.2.3, collection frequencies would be the usual thing to do when statistical language models are used, making it a *tf.icf* algorithm. One may have similar objections against the comparison of the language modelling approach with the probabilistic model. Figure 4.9 uses the probability of term importance and not the probability of term occurrence given relevance as used by the traditional probabilistic model of information retrieval.

Despite the differences, the similarity between the language models and the traditional models is important, because it gives insight in why *tf.idf* term weighting works and why the combination with relevance weighting, as e.g. done in the Okapi BM25 algorithm, works. Remember that, most weighting and ranking algorithms presented in section 2.4 are not so much based on models and theories, but instead on intuitions and on careful studies of the behaviour in test collections. The derivation presented above puts many of these intuitions in a different light. For instance, one of the original *tf.idf* intuitions, that term weights should be linear in the *tf* component, turned out to be not quite right in studies of Buckley, Allan, and Salton (1994) and Robertson and Walker (1994). The latter authors based their non-linear weighting of *tf* on an approximation of the 2-Poisson model, but the derivation above shows without any approximations that the *tf* component falls within the logarithm, making the term weights linear in $\log(tf)$. Another original intuition is that 'document length normalisation' should be applied to each term separately. Recent studies of Robertson and Walker (1994) and Singhal, Buckley, and Mitra (1996) showed the best performing algorithms only normalise each term partially. It is true that the denominator of the document term weight of figure 4.9 does contain the document length. The reason for this has nothing to do with the assumption that long and short documents are equally likely to be relevant, but instead it

is the result of proper normalisation of probabilities. Whether document length normalisation is applied or not is based on the use of either equation 4.3 or 4.6. The use of equation 4.6 will not have an impact on each query term, but instead it has an impact that is independent of the query length. If its impact were distributed over query terms as in the original intuition, then each term should indeed be normalised only partially.

This section supports the indications that the two old intuitions mentioned above were not quite right. Many other intuitions, like for instance the intuition that the use of document frequencies is essential for good retrieval (see section 4.2.4) or the intuition that probability of relevance estimation can be approximated by simple term occurrences for best match weighting algorithms (see section 4.4.5) might turn out to be not quite right as well.

4.7.3 A presence weighting algorithm for structured queries

The extended model introduced in section 4.3 can also be implemented as a presence weighting algorithm. This section shows how to rewrite the probability measures of the statistical translation retrieval model into a form that is similar to the basic model of section 4.2. Once the model is in this form, it is possible to follow the steps introduced in the previous section to derive the presence weighting algorithm. First, let's have a look again at the simplified notation of the statistical translation probability measure as introduced by equation 4.13:

$$P(D, S_1, S_2, \cdots, S_n) =$$
$$P(D) \prod_{i=1}^{n} \sum_{j=1}^{m} P(S_i | T_i = t^{(j)})((1 - \lambda_i) P(T_i = t^{(j)}) + \lambda_i P(T_i = t^{(j)} | D))$$

Filling in the estimators of equation 4.3, 4.4 and 4.5 results in the following formula. The probability measure $P(S_i | T_i = t^{(j)})$ will be replaced by the translation probability estimates $\tau_i(j)$.

$$P(D, S_1, S_2, \cdots, S_n) =$$
$$\frac{1}{N} \prod_{i=1}^{n} \sum_{j=1}^{m} \tau_i(j)((1 - \lambda_i) \frac{cf(t^{(j)})}{\sum_t cf(t)} + \lambda_i \frac{tf(t^{(j)}, d)}{\sum_t tf(t, d)})$$

The translation probabilities can be moved into the inner sum of unimportant and important terms. As summing is associative and commutative, it is not necessary to calculate each probability separately before adding them. Instead, respectively the collection frequencies and the term frequencies of the disjuncts can be added beforehand, properly multiplied by the translation probabilities. Only λ_i in the big sum is constant for every addition and can therefore be moved outside the sum, resulting in:

$$P(D, S_1, S_2, \cdots, S_n) =$$
$$\frac{1}{N} \prod_{i=1}^{n} ((1 - \lambda_i) \frac{\sum_{j=1}^{m} \tau_i(j) cf(t^{(j)})}{\sum_t cf(t)} + \lambda_i \frac{\sum_{j=1}^{m} \tau_i(j) tf(t^{(j)}, d)}{\sum_t tf(t, d)})$$

With these steps, the probability measure is rewritten back into its basic form, similar to equation 4.12. Following the exact same steps as in the previous section, the probability measure can now be rewritten into a presence weighting algorithm, resulting in equation 4.16.

$$
P(D, S_1, S_2, \cdots, S_n) \propto
$$
$$
\sum_{i=1}^{n} \log(1 + \frac{\lambda_i \left(\sum_{j=1}^{m} \tau_i(j) tf(t^{(j)}, d)\right) \sum_t cf(t)}{(1 - \lambda_i)(\sum_{j=1}^{m} \tau_i(j) cf(t^{(j)})) \sum_t tf(t, d)}) \tag{4.16}
$$

The model does not require the translation probabilities $\tau_i(j)$ to sum up to one for each i, since they are conditioned on the query term and not on the request word. Interestingly, for the final ranking it does not matter what the actual sum of the translation probabilities is. Only the relative proportions of the translations define the final ranking of documents. This can be seen by $\tau_i(j)$ which occurs in the numerator and in the denominator of the big fraction.

4.7.4 Discussion

Equation 4.16 relates to equation 4.15, the presence weighting algorithm of the basic model, as follows. Equation 4.16 sums up respectively the term frequencies and the collection frequencies of the possible translations of the words in the request weighted by the translation probabilities. If the sums are replaced by $tf'(t_i, d)$ and $cf'(t_i)$, that is:

$$
\begin{aligned}
tf'(t_i, d) &= \sum_{j=1}^{m} \tau_i(j) tf(t^{(j)}, d) \\
cf'(t_i) &= \sum_{j=1}^{m} \tau_i(j) cf(t^{(j)})
\end{aligned}
$$

then equation 4.16 equals equation 4.15. So, a weighted sum of respectively the term frequencies and the collection frequencies is used in a $tf.idf$-like ($tf.icf$ in this case) formula. If the translation probabilities are restricted to either 0 or 1, then the sums of respectively the term frequencies and the collection frequencies are 'real' sums, that is, no longer weighted sums. Translation probabilities might be restricted to 1 for possible translations if there is a deterministic process that converts query terms to one, and only one, request words. A stemmer is such a deterministic process: it converts a word always to the same stem, ignoring e.g. the word's part-of-speech. In these cases $tf(t_i, d) = \sum_j tf(t^{(j)}, d)$ and $cf'(t_i) = \sum_j cf(t^{(j)})$, which are exactly the same values for $tf(t_i, d)$ and $cf(t_i)$ if the deterministic process, like e.g. the stemmer, was used during indexing. So, for any deterministic process that is used during indexing, there is a corresponding query formulation strategy that produces the exact same ranking of the documents. This is not necessarily true if document frequencies are used instead of collections frequencies, because some possible translations might co-occur in one document.

Grouping morphological variants by using respectively the sum of the term frequencies and the sum of the document frequencies in a $tf.idf$ measure was done by Harman (1991) for an experiment with on-line stemming. The algorithm is implemented in the Inquery system as a synonym operator (Rajashekar

and Croft 1995). Harman introduced the algorithm because it provides a way to do experiments for a number of different stemmers, without the need to index the collection for each single experiment. Rajashekar and Croft introduced the algorithm because it is an intuitively plausible way to combine synonyms and synonym-like terms. Grouping possible translations of a source language term by the Inquery synonym operator has shown to be a successful approach to cross-language information retrieval (Ballesteros and Croft 1998; Pirkola 1998). This section derived a more general version of these algorithms from a formal model of information retrieval.

4.8 Two extensions: record fields and proximity

With the theory presented in this chapter, ranked versions of Boolean operators can be formulated by thinking of a suitable urn model. For instance, a suitable urn model for the query `(sustainable OR renewable) AND development` would be the following: The first term that is drawn from the relevant document is either "sustainable" or "renewable". The second term that is drawn from the document is the term "development". Likewise, it is possible to find suitable urn models for the Boolean proximity operators of section 3.3.2 and the field searches of section 3.3.5.

4.8.1 Three -or more- levels of importance

Field searches are introduced by extending the model presented so far by allowing the random variable I_i to have more than two realisations. If words from e.g. the title field are searched, before drawing a term it is decided if the term is drawn from the entire collection ($I_i = 0$), the relevant document ($I_i = 1$), or from the relevant document's title ($I_i = 2$). If the simplified notation μ_i is used instead of $P(I_i\!=\!2)$ and $P(T_i|F, D)$ instead of $P(T_i|I_i\!=\!2, D\!=\!d)$, then the resulting simplified notation of the basic field search measure is as follows.

$$P(D, T_1, T_2, \cdots, T_n) = \\ P(D) \prod_{i=1}^{n}((1\!-\!\lambda_i\!-\!\mu_i)P(T_i) + \lambda_i P(T_i|D) + \mu_i P(T_i|F, D)) \qquad (4.17)$$

Similarly for proximity searching, before drawing a term first it is decided if the term is drawn from the entire collection ($I_i = 0$), the relevant document ($I_i = 1$), or from the collection of terms in the relevant document that have a proximity relation with a previously drawn term ($I_i = 2$). Except for the different sample space of each I_i, the probability measures of equation 4.2 and 4.10 have to be extended because the query terms are no longer independent. Dependence relations between terms that are adjacent in the query are now permitted. This is visualised in the Bayesian Network of figure 4.10. If the simplified notation μ_i is used instead of $P(I_i\!=\!2)$ and $P(T_i|T_{i-1}, D)$ instead of
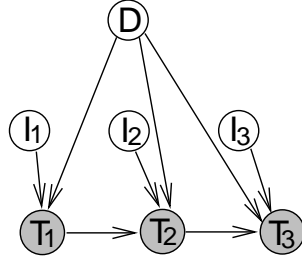
Figure 4.10: Graphical model of dependence relations between query terms

$P(T_i|T_{i-1}, I_i=2, D)$, then the resulting simplified notation of the basic proximity searching measure is as follows.

$$
\begin{aligned}
P(D, T_1, T_2, \cdots, T_n) \;=\; & P(D)\left((1-\lambda_1)P(T_1) + \lambda_1 P(T_1|D)\right) \\
& \prod_{i=2}^{n}\left((1-\lambda_i-\mu_i)P(T_i) + \lambda_i P(T_i|D) + \mu_i P(T_i|T_{i-1}, D)\right)
\end{aligned}
\tag{4.18}
$$

Likewise, it would be possible to introduce a fourth level of importance: one that uses information on multiple fields or query terms like e.g. trigram probabilities. Statistical translation can be included by assuming that the translation of a term is done independently of relevance, importance and independently of the database field or the previous term(s).

4.8.2 Field searches

For the example of searching documents whose title contain the query terms, the probability of drawing a term from the title may simply be defined by the number of occurrences of that term in the title, divided by the length of the title.

$$
P(T_i{=}t_i|F, D{=}d) \;\;=\;\; \frac{\#(t_i \text{ in TITLE of } d)}{\sum_t \#(t \text{ in TITLE of } d)}
\tag{4.19}
$$

4.8.3 Adjacent terms

A suitable urn model for the adjacency operator would be the following. First a term is drawn from the relevant document like presented earlier. Then a second term is drawn from the collection of terms that are adjacent to any occurrence of the first term in the relevant document. The estimation of probabilities is straightforward. Whatever the proximity operator, the number of hits should be counted and normalised properly. For the adjacency operator, familiar bigram estimates are used.

$$
P(T_i{=}t_i|T_{i-1}{=}t_{i-1}, D{=}d) \;\;=\;\; \frac{\#(t_{i-1} \text{ ADJ } t_i \text{ in } d)}{tf(t_{i-1}, d)}
\tag{4.20}
$$

Bigram probabilities were also used by Miller, Leek, and Schwartz (1999) and Song and Croft (1999). If all of the occurrences of t_{i-1} have an adjacent term t_i in k then the probability is one. If none of the occurrences of t_{i-1} have an adjacent term t_i in k then the probability of the adjacent operator is zero.

4.8.4   Near terms

A suitable urn model for the NEAR operator would be the following. First a term is drawn from the relevant document as it was done in the simple model. Then a second term is drawn from the collection of terms that are within a window of x terms to any occurrence of the first term in the relevant document. The resulting estimator is shown below.

$$P(T_i = t_i | T_{i-1} = t_{i-1}, D = d) \quad = \quad \frac{\#(t_{i-1} \text{ NEAR } t_i \text{ within } x \text{ in } d)}{2\, x\ tf(t_{i-1}, d)} \qquad (4.21)$$

This operator matches terms in a specified window of x terms in any order. Alternatively, the operator could for instance be specified in such a way that it matches terms in a specified order. In this case, the actual window is twice as small and the $2x$ in the denominator should be replaced by x.

4.8.5   Relation to strict Boolean searching

The relation of the extensions introduced above with their strict Boolean versions is as follows. If a value of 1 is assigned to the probability of importance level 2 given relevance, i.e. $\mu_i = 1$, then the formula assigns zero probability to every document that does not exactly match the structured query. In the case of a search for query terms in the title field, the system only retrieves documents in which the term occurs in the title; other documents are assigned zero probability. In case of proximity operators, the system assigns zero probability to document that do not contain the term adjacent to, or near to, the query term on position $i - 1$.

4.9   Conclusion

This chapter introduced a language model-based approach to information retrieval consisting of a basic model for the processing of simple queries and an extended model for the processing of structured queries. Proximity operators and field search operators can be included by introducing more than two levels of importance, that is, by introducing a mixture of more than two models.

   The combination of evidence from different sources of information should be done by the following recipe. Different representations of the document can be combined by using a mixture of more than two models of the document. Different representations of the request or information need can be combined by using the statistical translation model (query formulation model). The optimum way to combine all the evidence can be found by using the EM-algorithm on some previously found examples of relevant documents.

Chapter 5

Experimental results

Using the evaluation methodology described in appendix A, this chapter reports on the evaluation of a language model-based retrieval system. Section 5.1 introduces three basic retrieval tasks to be evaluated. Section 5.2 determines an optimum setting of the model on the Cranfield test collection. In section 5.3, these settings are used in three experiments that compare the performance of the language model-based retrieval algorithms with the performance of traditional retrieval models and today's top performing term weighting algorithms.

5.1 Introduction

This chapter will evaluate the new retrieval model by comparing it to some well-established methods in a controlled experiment. Traditionally there are three quite distinct problems that retrieval models try to solve. The three problems, which were introduced in chapter 2, are the following:

1. term weighting and ranking algorithms;
2. relevance feedback from examples of relevant documents;
3. structured queries.

Following the three problems, three basic retrieval tasks were designed. The three experiments serve to illustrate that the language model-based system performs well in situations that call for, respectively, the ability to rank documents without the use of relevance information, the probability of relevance estimation from relevant documents, and the ability to process Boolean-structured queries. The first experiment is set up as the ad-hoc task in TREC. The ad-hoc task represents the situation in which a user enters a query that is previously unseen by the system. The second task is called the retrospective relevance weighting task. It determines the ability of the retrieval algorithms to re-estimate their parameters from all known relevant documents. This task is mainly of theoretical interest and was done before by Robertson and Sparck-Jones (1976) and Sparck-Jones, Walker, and Robertson (2000) for the probabilistic model.

The third task measures the ability of systems to process manually formulated Boolean-structured queries.

For each of the three retrieval tasks, the language model-based system will be compared with one or more of the traditional models that try to solve the problem associated with the task. Ideally, we would like to take the three classical models of information retrieval for comparison, respectively: the Boolean model, the vector space model and the probabilistic model. Since the Boolean model does not provide a ranking of the documents, the popular p-norm model will be used instead. Of the vector space model and the probabilistic model, two versions will be evaluated for comparison. Of each model, one version represents the model as it was introduced in the 1970's and one version represents the model as it was actually used in the TREC experiments of the late 1990's.

The chapter is organised as follows. Section 5.2 reports on preliminary tests on the Cranfield collection that are used to determine the best version of the model and the best value of the unknown parameter λ. Section 5.3 reports the performance results of the language model and the well-established models on the three tasks introduced above, respectively the ad-hoc retrieval task, the retrospective relevance weighting task and the manually formulated Boolean queries task. Section 5.4, reports on a few post-hoc tests to check if the decisions made from the Cranfield results were reasonable. Finally, section 5.5 draws conclusions and identifies the additional experiments of the chapters 6 and 7.

5.2   Determining the model's optimum setting

The goal of the experiment described in this section is not to test a hypothesis, but to choose a reasonable version of the model and to establish a reasonable value of the unknown parameter λ. This is usually called 'tuning' of a model. In this section, results are reported of tuning the model on the Cranfield test collection (Vickery 1970). The Cranfield collection is a small collection of 1398 abstracts on aerodynamics with 225 requests. For retrieval standards, the number of documents is really small. The collection's advantage is the relatively large number of requests, and the fact that all documents have been judged for each request.

The tests were done with the experimental language model retrieval engine developed at the University of Twente. Documents and queries were preprocessed as follows. Tokenisation was done by assuming that all non-letters, including hyphens, are word boundaries. Words occurring in the Smart stop list (Smart 1994) were removed. The remaining words were stemmed using the Porter stemmer (Porter 1980).

5.2.1   Exploring four ways of specifying the probabilities

Chapter 4 introduced the definitions of the three probability measures $P(D)$, $P(T_i|D)$ and $P(T_i)$ by equations 4.3, 4.4 and 4.5. Of the first and the last of these measures, an alternative definition was introduced by equations 4.6 and

4.7. This leads to four versions of the model that have to be explored. For completeness the weighting algorithms of the four versions are displayed below. Remember that the sum of $i = 1$ to n covers the query terms on each position i, which recomputes the weight of duplicate terms. In practice, this might of course be implemented by multiplying the weight of the term by the frequency of occurrence of the term in the query.

$$\text{score}_1(d) \;=\; \sum_{i=1}^{n} \log(1 + \frac{\lambda_i \, tf(t_i, d)(\sum_t cf(t))}{(1 - \lambda_i) cf(t_i)(\sum_t tf(t, d))})$$

$$\text{score}_2(d) \;=\; \sum_{i=1}^{n} \log(1 + \frac{\lambda_i \, tf(t_i, d)(\sum_t df(t))}{(1 - \lambda_i) df(t_i)(\sum_t tf(t, d))})$$

$$\text{score}_3(d) \;=\; \log(\sum_t tf(t, d)) + \sum_{i=1}^{n} \log(1 + \frac{\lambda_i \, tf(t_i, d)(\sum_t cf(t))}{(1 - \lambda_i) cf(t_i)(\sum_t tf(t, d))})$$

$$\text{score}_4(d) \;=\; \log(\sum_t tf(t, d)) + \sum_{i=1}^{n} \log(1 + \frac{\lambda_i \, tf(t_i, d)(\sum_t df(t))}{(1 - \lambda_i) df(t_i)(\sum_t tf(t, d))})$$

The four versions differ by the use of document frequencies instead of collection frequencies for the versions 2 and 4, and by a document length correction value which is added for the versions 3 and 4.

5.2.2 Determining a value for λ

If there is no previous relevance information available for a query, i.e. none of the relevant documents has been identified yet, each term that is entered by the user and that is not in the stop list, will be considered equally important. As shown in chapter 4, the model has only one unknown parameter in this case, because λ_i will be equal for each position i in the query. The unknown parameter will simply be called λ in the following. One way to find an optimum value for λ is to take a test collection and evaluate retrieval performance for a wide range of values of λ.

Figure 5.1 shows the average precision averaged over 225 queries plotted against different values of λ of the four versions. The figure shows similar shapes of the plots of the four versions of the model. Necessarily, for $\lambda = 0$, the system will produce random results and perform close to zero average precision. When λ is increased the average performance will improve to a maximum. If λ is increased further, the average performance will steadily decline. The versions that use document frequencies outperform the versions that use collection frequencies. Document length correction does not really improve version 1 that uses collection frequencies, but it shows a slight improvement of version 3 that uses document frequencies. The original version of the model, version 1, reaches its maximum performance of 0.416 for $\lambda = 0.35$. Miller et al. (1999) report a value of $\lambda = 0.3$ for this model on the TREC collection. The version using both alternative probability specifications, version 4, is the best performing version.
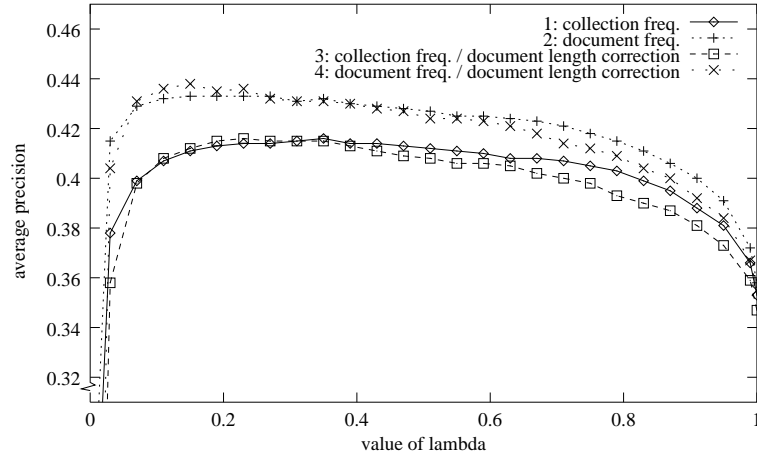
Figure 5.1: Average precision against values of λ on Cranfield

It reaches an average precision value of 0.437 for $\lambda = 0.15$, which was reported before in (Hiemstra and Kraaij 1999).

5.2.3   A prediction interval for λ?

For version 4 of the model a value of $\lambda = 0.15$ will on average produce maximum performance in terms of average precision. This raises the question: "how reliable is this value?" A single query will usually not perform maximally for exactly $\lambda = 0.15$, but it might be possible to define an interval for λ in which most queries reach optimum performance. As it turns out, the smallest 95 % prediction interval of λ for version 4 of the model would be $[0.03, 1.00]$. So, 95 % = 214 of the 225 Cranfield queries reach optimum performance between $\lambda = 0.03$ and $\lambda = 1.00$. Apparently, there is a huge variation in the optimum value of λ when single queries are considered.

Averaging the performance of 225 queries hides many of the interesting details of the behaviour on single queries. Just as recall-precision graphs of single queries show much more chaotic behaviour than the average graph of all queries, so do the graphs of the average importance λ against average precision. All examples presented in this section are the result of version 4 of the model.

A common picture is one that is similar to the average performance over all queries. First the graph reaches an early maximum, after which the graph monotonically decreases with steps. An example is shown in figure 5.2. Most Cranfield queries show similar behaviour.

Another typical example shows the opposite behaviour. The performance increases slowly and optimum performance is only reached when λ is close to one, that is when all query terms are almost with certainty important. These queries would probably show quite good performance in a traditional exact match retrieval system. Note that the plots have different vertical scales. The

absolute performances of the examples differ considerably, but at this point only the shapes of the plots are of interest.
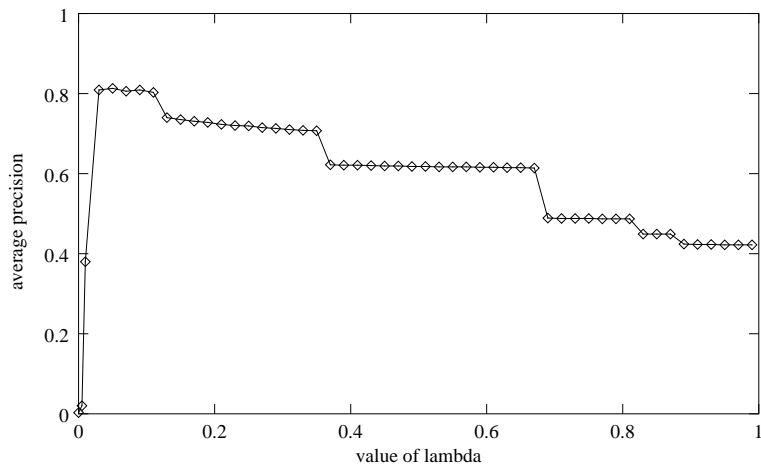
Figure 5.2: Example of predominantly unimportant terms (query 9)
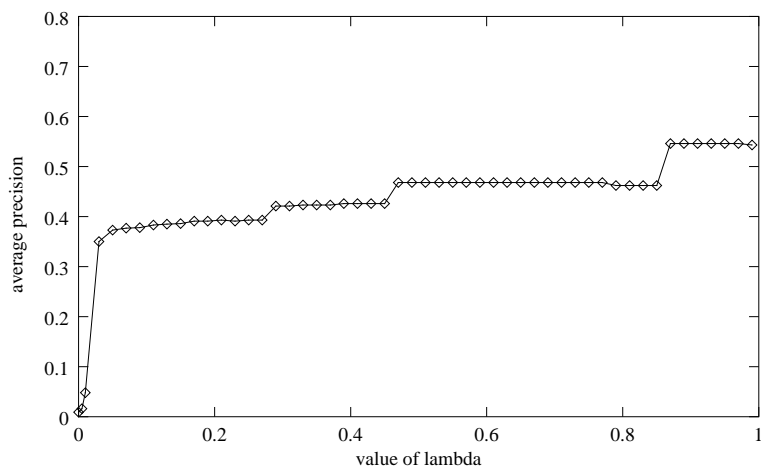
Figure 5.3: Example of predominantly important terms (query 18)

Some queries do not follow any of the two patterns shown above. Many show a large number of local maxima. The example plot of figure 5.4 has at least four local maxima. The global maximum occurs for a really precise value of λ on a really small interval. Without relevance information it is impossible to predict where the maximum will exactly fall. Apart from the very hard queries, there are also a number of queries that are plain simple. The example of figure 5.5 shows a query that performs extremely well for any $\lambda > 0$.
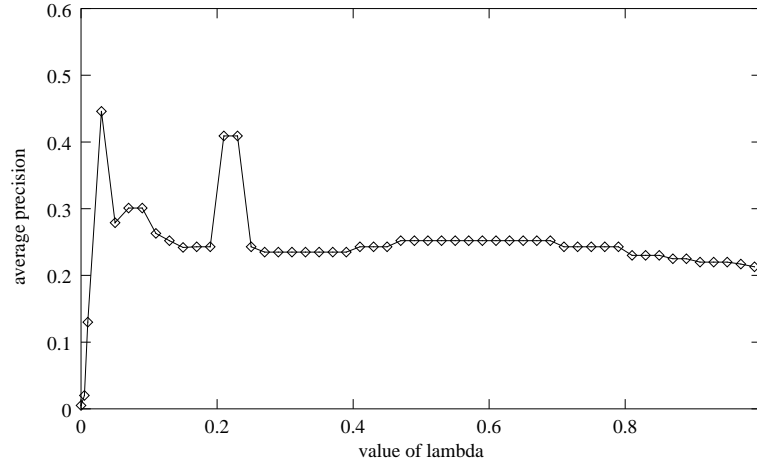
Figure 5.4: Example of 'unpredictable behaviour' (query 36)
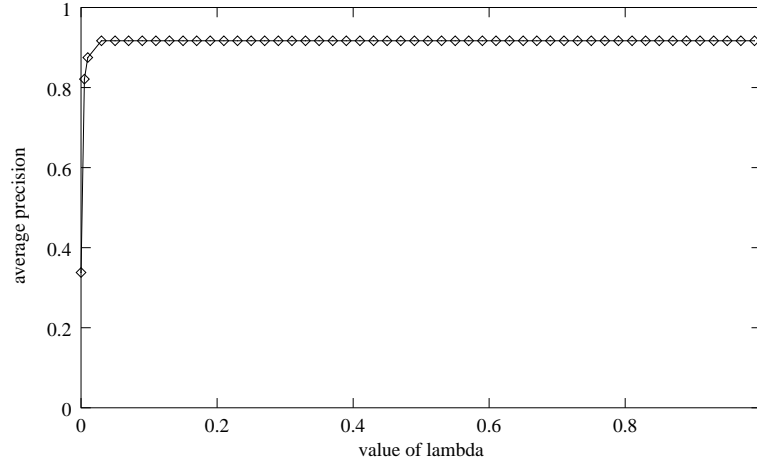
Figure 5.5: Example of 'easy retrieval' (query 78)

The examples presented in this section show that different queries need different retrieval strategies. Because of this Ng (2000) used a two stage retrieval method. In the first stage, the top 5 of the retrieved documents is used to estimate a query specific value of λ. This weight is used in a second stage that produces the actual retrieval results. For the ad-hoc experiments in sections 5.3.2 a one stage retrieval process will be used to make the comparison with other models not too complicated. All queries will use the same importance weight λ. The fact that different queries need different strategies does not diminish the fact that on average only 15 % of the query terms is important. This implies for instance that a query of length four will not have any important term in over half ($0.85^4 = 0.522$) of the relevant documents.

5.2.4 Choosing a test system

This section reports on the performance of four different versions of the language model-based retrieval system on the Cranfield collection for a wide range of values for the unknown parameter λ. The best performance was reached by version 4 for $\lambda = 0.15$. It is not clear that this setting will also be the best on the TREC collection. The TREC collection contains very different documents and is about 400 times as large as Cranfield. It would be interesting to do a comparison of the same scale as reported above on a larger test collection that is more similar to the TREC collection. Lacking such a comparison, the experiments described in the remainder of this chapter will use version 4 and $\lambda = 0.15$. The same system setting was used in a number of other experiments (Hiemstra 1998a; Hiemstra and Kraaij 1999; Hiemstra 2000).

5.3 Evaluation results

This section presents the results of version 4 of the model on three different tasks. The first task is the ad hoc task which represents the situation of a user who enters a previously unseen query and then checks the results. The second task is the 'retrospective relevance weighting' task, which measures the ability of the system to estimate optimal term weights from examples of relevant documents. The third task measures the system's ability to rank documents if the user enters a Boolean-structured query.

Experiments were done using the Mirror DBMS, a prototype database management system especially designed for multimedia and web retrieval (De Vries 1999). The Mirror DBMS combines content management and data management in a single system. The main advantage of such integration is the facility to combine information retrieval with traditional data retrieval. Furthermore, information retrieval researchers can experiment more easily with new retrieval models, using and combining various sources of information. This is an important benefit for advanced information retrieval like for instance web retrieval, speech retrieval, and cross-language retrieval. Each of these might require the use of several representations of content, which is hard to handle in the traditional file-based approach, and becomes too slow in traditional database systems.

The TREC collection that is used for the experiments consists of a total of 528,024 documents from four separate sources: Federal Register, Los Angeles Times, Foreign Broadcast Information Services and the Financial Times. Documents and queries were preprocessed as for the Cranfield experiments of section 5.2 using the title and description field of the TREC topics. Additionally, words that are specific to the TREC domain, like "document" and "relevant" were stopped from the topics. The same index was used for all three experiments.

5.3.1 Comparing results of two algorithms

Pair-wise comparisons between runs are based on the average precision measured over the ranks of relevant documents as described in section A.3. The average

precision at 11 levels of recall will be reported by recall-precision plots. The appendix gives a detailed description of the results on other measures. The two-tailed pair-wise sign test, which is described in section A.4, will be used to determine significant differences between two runs. Differences at the 5 % level are reported as significant. In this case, the critical value is 17, that is, the number of times that the least frequent sign occurs should be 17 or less than 17 in order for the difference to be significant. The appendix also reports differences that are significant at the 1 % level, for which the critical value is 15.

5.3.2   Results on the ad hoc task

The first experiment is a TREC-style automatic ad-hoc experiment using TREC topics 401-450. It serves to illustrate that the language model-based system performs well on a task where no relevance information is available and the system has to rely on the similarity between the query and the documents. The experiment compares the average precision of five different term weighting algorithms that were presented in section 2.4. The weighting algorithms implemented and tested are the original *tf.idf* with cosine normalisation, the Robertson/Sparck-Jones weight of the traditional probabilistic model, the Lnu.ltu formula and the BM25 formula. The Lnu.ltu slope parameter was set to 0.2. The BM25 tuning parameters were set to $k_1 = 1.2$, $b = 0.75$ and $k_3 = \infty$. The values of tuning parameters of the weighting formulas are the ones reported in respectively (Singhal et al. 1996) and (Robertson et al. 1999).[1]
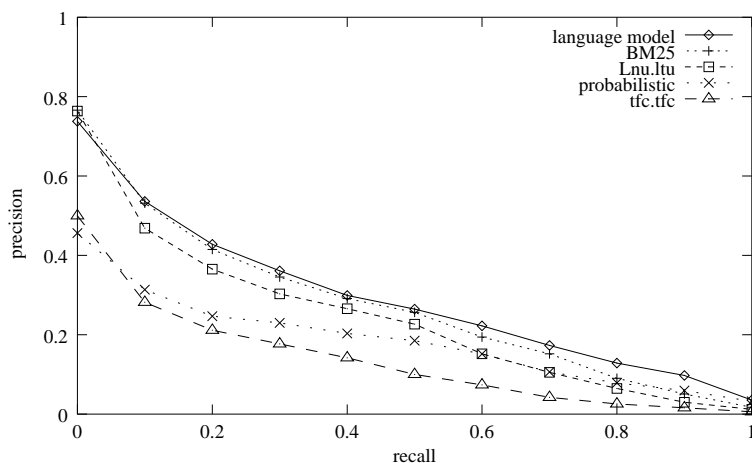
Figure 5.6: Recall-precision plots of ad-hoc runs

The experiment shows that both the original probabilistic model and the original vector space model underperform on this task. The Lnu.ltu, BM25 and

---

[1]The results for BM25 differ from (Hiemstra and De Vries 2000), which used $k_1 = 2$ reported in (Robertson and Walker 1994).

language model algorithms perform better. In fact, they show similar results on the very high precision / 0.0 recall point, where Lnu.ltu and BM25 are slightly better than the language model. On the other recall points, the performance of the language model and BM25 diverges from the performance of Lnu.ltu. Both the language model and BM25 seem to perform consistently better than the Lnu.ltu algorithm on these points. The precision values at document cut-offs 10, 30 and 100 and the average precision over the ranks of relevant documents retrieved are displayed in table 5.1. In the table "LM" stands the for language model-based algorithm.

run	precision at document:			average precision
	10	30	100	
tfc.tfc	0.240	0.187	0.122	0.126
probabilistic	0.248	0.187	0.153	0.165
Lnu.ltu	0.450	0.345	0.214	0.229
BM25	0.484	0.366	0.234	0.261
LM	0.494	0.385	0.235	0.277

Table 5.1: Results of ad hoc queries

If the average precision is taken as the measure to base our hypotheses upon, then the following conclusions can be drawn. The difference between the language model and BM25 is not statistically significant. The difference between the language model and Lnu.ltu and the difference between BM25 and Lnu.ltu are both significant. The difference between any of the three modern term weighting algorithms and the original $tf.idf$ algorithm and the traditional probabilistic model are also significant. Interestingly, despite the big absolute difference between the average precision of the probabilistic model and the vector space model with original $tf.idf$ weights, this difference is not statistically significant.

5.3.3 Results of relevance weighting

The second experiment takes the relevant documents of each topic (401-450) to estimate relevance weights, which are used retrospectively to determine optimal performance on the collection. The same experiment was done by Robertson and Sparck-Jones (1976) on the Cranfield collection using the traditional probabilistic model, and by Sparck-Jones et al. (2000) on the TREC collection using the traditional probabilistic model and the BM25 algorithm. The purpose of this experiment is two-fold. Firstly, the experiment shows how the language model's relevance weighting method performs compared to relevance weighting of the traditional probabilistic model and the BM25 formula. Secondly, by comparing the performance with the experiments presented in the previous section, the experiments show how much can be gained if the system has perfect knowledge about the distribution of terms over relevant and non-relevant documents.
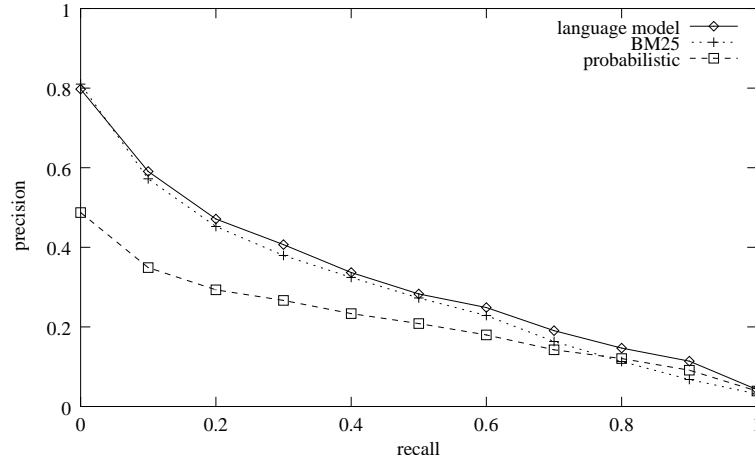
Figure 5.7: Recall-precision plots of retrospective relevance weighting

Comparison of figure 5.6 and figure 5.7 shows that the three models show similar increase in performance, except for the probabilistic model's increase of precision for high recall levels. On the 0.8 and 0.9 recall levels the traditional probabilistic model noticeably outperforms the BM25 model. The precision at document cut-offs 10, 30 and 100, and the average precision measure is displayed in table 5.2.

run	precision at document:			average precision
	10	30	100	
probabilistic	0.268	0.217	0.172	0.198
BM25	0.526	0.400	0.258	0.289
LM	0.550	0.410	0.260	0.311

Table 5.2: Results of retrospective relevance weighting

Pair-wise comparison of the average precision of the experiments shows the following. The BM25 algorithm and the language model-based algorithm perform significantly better than the probabilistic model on this task. There is not enough evidence to disprove equal performance of the BM25 algorithm and the language model-based algorithm for relevance weighting.

Comparison of the retrospective weighting experiments and the ad-hoc experiments reveals the following. The average precision after retrospective relevance weighting is significantly better than the ad-hoc versions of the traditional probabilistic model and the language model. There is not enough evidence to show that relevance weighting improves the performance of the BM25 algorithm.

The query by query comparison shows that the three methods actually decrease the average precision of respectively 4, 18 and 10 out of 50 queries. This seems rather alarming, because a good relevance feedback method should never decrease performance if the weights are used retrospectively. For the language model, we do have a clue why the relevance weighting algorithm seems to be suboptimal. As said in section 4.4, the likelihood criterion of the EM-algorithm is not directly related to the aim of optimising the probability of relevance, so it might not lead to it. The BM25 algorithm makes more mistakes still, some in fact quite substantial. This suggests that it might be possible to improve upon the relevance weighting algorithms of the language model and the BM25 algorithm. More research into relevance feedback algorithms is therefore needed.

5.3.4 Results on Boolean-structured queries

The third experiment uses manually formulated Boolean queries. For this experiment we used the Boolean queries that were formulated by Schiettecatte (1998) for TREC topics 301-350. Wildcards and multi-term expressions were replaced by Boolean equivalents, using the OR-operator for wild cards and the AND-operator for multi-term expressions. The experiment tries to answer two questions. First of all it shows how the language model-based system performs compared to a system based on the p-norm model. Secondly, it measures the additional benefit of extended Boolean models over versions of the model that do not use the Boolean operators.

Following the experiments reported by Salton et al. (1983) binary query weights and *tf.idf* document weights were used for the p-norm experiments. Experiments were done both using tfc weights and Ltu weights for documents. The p-norm model can be reduced to the vector model by assigning a value of 1 to both p parameters. Using a higher value for p, say $p = 2$, should therefore show improved results. The language model does not have a similar knob. Therefore one experiment 'LM vector' was done after throwing away the Boolean operators, just leaving the terms. Again, using the Boolean operators as in 'LM Boolean' should show improved results compared to not using them. For the language model-based algorithm, queries were converted automatically to their conjunctive normal form. Boolean expressions that did not contradict on the number query positions, like for instance the disjunction of two two-word phrases as in `(funny AND tables) OR (amusing AND chairs)` were also converted to their conjunctive normal form. These queries deserve additional attention in future evaluations.

The experiment shows that not much can be gained by the special treatment of Boolean operators. Special treatment of Boolean operators seems to have the same absolute impact on the p-norm model as on the language model: about 0.02 gain in average precision. The improvement of performance between $p = 1$ and $p = 2$ of the p-norm model is significant if tfc weights are used, but the improvement is no longer significant if the p-norm model uses Ltu weights. The difference between the 'LM Boolean' run and the 'LM vector' run is also not statistically significant.
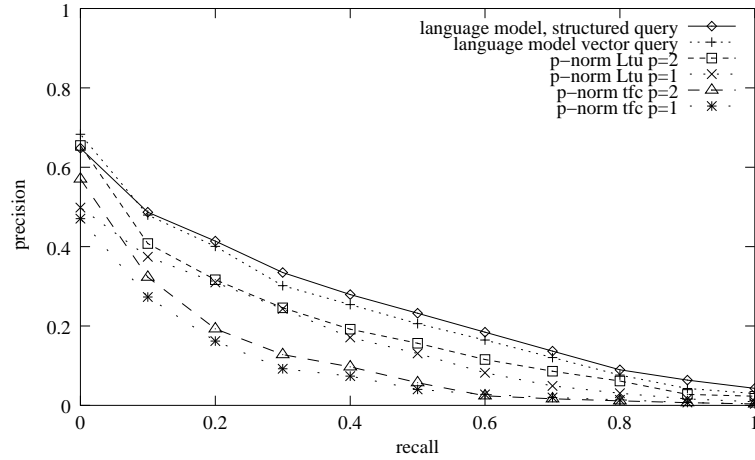
Figure 5.8: Recall-precision plots of structured queries

run	precision at document:			average
	10	30	100	precision
p-norm tfc p=1	0.222	0.140	0.088	0.084
p-norm tfc p=2	0.286	0.171	0.103	0.102
p-norm Ltu p=1	0.276	0.223	0.147	0.156
p-norm Ltu p=2	0.366	0.260	0.180	0.182
LM vector	0.398	0.303	0.185	0.224
LM Boolean	0.372	0.292	0.188	0.244

Table 5.3: Results of Boolean-structured queries

Term weighting seems to have a bigger impact on retrieval performance. The difference between the Ltu $p = 2$ experiment and the tfc $p = 2$ experiment is statistically significant. The difference between the 'LM Boolean' run and the Ltu $p = 2$ run is also significant.

5.4   Some reflection on the alternative versions

From the results on the Cranfield collection, it was hypothesised in section 5.2.1 that version 4 of the model would be the preferred version for further inspection. This section explores if this hypothesis was correct by reporting on the results of the algorithm without the document correction component (version 2) and the results of the algorithm using collection frequencies instead of document frequencies (version 3).

5.4.1 Document length correction

Table 5.4 displays the average precision measures of the language model experiments presented above if document length correction was not used. Version 2 of the model uses a uniform marginal probability of relevance of a document. Comparison of the results given in table 5.4 with the results given in table 5.1,

run	precision at document:			average
	10	30	100	precision
LM ad-hoc	0.474	0.354	0.223	0.263
LM rel. weights	0.514	0.400	0.251	0.299
LM vector	0.380	0.299	0.184	0.216
LM Boolean	0.384	0.301	0.196	0.234

Table 5.4: Results of using LM version 2

5.2 and 5.3, indicates that the use of a uniform marginal probability of relevance of a document performs noticeably worse than the alternative used in the experiments reported above. The differences with the version 4 runs are significant for ad-hoc run and the relevance weighting runs. The differences with the version 4 runs of the Boolean-structured queries are however not significant.

5.4.2 Collection vs. document frequencies

Table 5.5 presents the results of the experiments presented above if collection frequencies instead of document frequencies are used (version 3). Comparison of

run	precision at document:			average
	10	30	100	precision
LM ad-hoc	0.490	0.372	0.230	0.273
LM rel. weights	0.542	0.408	0.262	0.309
LM vector	0.376	0.274	0.175	0.221
LM Boolean	0.372	0.284	0.181	0.240

Table 5.5: Results of using LM version 3

the results given in table 5.5 with the language model results given in table 5.1, 5.2 and 5.3, indicates that the use of collection frequencies (equation 4.5) instead of document frequencies (equation 4.7) has the tendency to perform a little bit worse (about 1 % on all four runs). For all four runs, the differences with the version 4 runs is not significant. Whether the difference is significant or not, the slightly worse results of collection frequencies seems to be consistent over the difference between the query sets of the ad-hoc and relevance weighting runs compared to the Boolean-structured query runs. However, the use of collection

frequencies instead of document frequencies is not nearly as bad as generally assumed (see e.g. Church and Gale 1999).

5.5   Discussion

In this chapter the performance of the new language model-based retrieval system was evaluated against the performance of systems based on three traditional models of information retrieval: the vector space model, the probabilistic model and the p-norm model. On the three tasks and on the test collections and test queries used, the new model's average precision is better than the average precision of the traditional models. All these differences are significant at the 5 % level, except for the difference with the BM25 algorithm for which the test was not able to detect a significant difference.

The results on the ad-hoc queries show that the new model performs well on the main TREC task. This is rather impressive if one recalls that for instance the Lnu.ltu term weighting algorithm is the result of many years of research within the Smart projects in which hundreds of term weighting algorithms were tried (see section 2.4). Section 4 showed that the language model's term weighting algorithm is completely defined by the underlying theory. This is not the case for the modern term weighting algorithms, like Lnu.ltu and BM25, but until recently these algorithms did have the advantage that they simply performed better than simple or well-motivated algorithms.

The relevance weighting experiment presented in section 5.3.3 shows that, if all relevant documents are known, the language model shows a performance gain that is similar to the gain of the traditional probabilistic model and better than the performance gain of the BM25 algorithm. The retrospective relevance weighting experiment serves no other purpose than just that. There is little practical use for an algorithm that needs to know all relevant documents beforehand. There is however use of relevance weighting algorithms that can predict the relevance of future documents from a small sample of known relevant documents. In chapter 7, the usefulness of the relevance weighting algorithm will be further explored in a prototype adaptive filtering system that uses the user's feedback to predict the relevance of future documents.

The experiment with Boolean-structured queries showed some improvement over unstructured queries, but the results are not significant. Actually, the use of traditional Boolean-structured queries is somewhat unnatural for the language model-based system, because the queries need to be converted automatically to their conjunctive normal form before processing. One of the modern query languages for structured queries as presented in section 3.3.4 would be more suitable. A second interesting application of structured queries might be automatic query formulation and expansion by the system. A promising application in this respect is that of cross-language information retrieval, in which a request in one language is used to formulate a structured request in another language automatically. In chapter 6, the usefulness of this application is further explored by using automatically translated queries from machine readable dictionaries.

Chapter 6

Cross-language information retrieval

This chapter reports on the evaluation of a prototype cross-language information retrieval system. The chapter's key issue is the question whether it is possible to improve upon a system that uses serious disambiguation methods during translation by using the structured query approach introduced in chapter 4. A brief introduction to cross-language retrieval is given in section 6.1. Section 6.2 explores possibilities for the comparison of the document translation approach with the query translation approach. Section 6.3 introduces three basic methods for query translation. Section 6.4 addresses heuristics and statistics for disambiguation when the query translation approach to cross-language retrieval is applied. Section 6.5 discusses experimental setup and experimental results. Finally, section 6.6 contains concluding remarks.

6.1   Introduction

Cross-language retrieval supports the users of multilingual document collections by allowing them to submit queries in one language, and retrieve documents in any of the languages covered by the retrieval system. Consider the example of Dutch queries on an English document collection. Cross-language retrieval can be achieved by: *off-line document translation*: translating English documents into Dutch, then indexing in Dutch; *off-line index translation*: indexing English documents in English, then translating the resulting index into Dutch; *on-line query translation*: indexing English documents in English and translating Dutch queries on the fly into English. The latter method is preferred when the former two are impractical. Query translation is enforced in environments where it would be impossible to produce translations for all documents in the document base and/or translated indices for each language. Document translation has the major advantage that it is possible to present the user a high quality preview of the retrieved documents. Translating documents after they are retrieved,

as offered by some web search engines, does not suffice because it does not help users to identify material that they might want to have translated. Since it presupposes that the user has already found the relevant document in its original foreign language, it fails to support exactly that part of a search in a multilingual environment which is the most difficult one: to formulate a query which will take the user to the foreign language document of interest.

6.1.1   Disambiguation strategies

If a word has more than one possible translation it is called ambiguous, e.g. the English word "plant" has two possible French translations "plante" for the sense of 'vegetation' and "usine" for the sense of 'factory'. The term 'disambiguation' is used in two ways in this chapter. Disambiguation might refer to the process of choosing one best translation, which is called explicit disambiguation. Disambiguation might also refer to the estimation of probabilities for each possible translation: implicit disambiguation. The disambiguation process might for instance assign a probability of 0.8 to `plante` and 0.2 to `usine`. The probabilities can be used to identify the most probable translation explicitly, but, if the query translation approach is taken, they might also be used implicitly during retrieval by weighting each possible translation with the methods described in chapter 4. In the Twenty-One project, translation and disambiguation can be pursued in four ways:

1. Using existing machine translation software (LOGOS);
2. selection of the preferred translation from a machine readable dictionary (Van Dale);
3. using domain specific dictionaries that are automatically generated on the basis of statistically processed parallel corpora;
4. disambiguation on the basis of the frequency of noun phrases in the document collection.

Twenty-One is a project funded by the EU Telematics programme, sector Information Engineering that started in 1996 and was completed in 1999. The project had three important focal points. Firstly, the project has a clear target domain, focusing on disclosing literature on sustainable development in four languages: Dutch, English, French and German. Secondly, it has a strong focus on the disclosure of paper documents which have to be scanned and converted to an electronical format by optical character recognition software. The third focus is on natural language processing and cross-language retrieval in the four supported languages. At indexing time, noun phrases are recognised and used as complex index terms. As the Twenty-One domain is limited and as heavy pre-processing and storage of scanned documents has to be reckoned with anyhow, it is a classic case for the document translation approach. Document translation using existing machine translation software is the approach taken by the Twenty-One demonstrator system, which was the first on-line text retrieval system supporting cross-language search in Europe (Twenty-One 1998). The other

three approaches mentioned above were developed within the project as well. They are evaluated in this chapter.

6.1.2   A model of cross-language information retrieval

One might argue that the new model of information retrieval that is introduced in chapter 4 is specially designed for cross-language information retrieval. The model explicitly includes statistical translation, which is used in chapter 5 to process manually formulated Boolean-structured queries. In this chapter, the model will be used to process structured queries that are automatically generated by the translation tools mentioned above. The tools will generate queries in a convenient conjunctive normal form, and include the translation probabilities.

This chapter further investigates the new model's ability to use structured queries. It tries to answer the question whether it is possible to improve upon a system that uses sophisticated explicit disambiguation methods during translation, by using the structured query approach introduced in chapter 4. The disambiguation strategies are provided by the modules that are developed within the Twenty-One project. The experiments reported in this chapter were done as part of the Twenty-One project and were published before as (Hiemstra and de Jong 1998 and 1999). The chapter is organised as follows. d Section 6.2 explores possibilities for the comparison of the document translation approach with the query translation approach. Section 6.3 introduces three basic methods for query translation. Section 6.4 addresses heuristics and statistics for disambiguation if the query translation approach to cross-language retrieval is applied. Section 6.5 discusses the experimental setup and experimental results. Finally, section 6.6 contains concluding remarks.

6.2   Document translation vs. query translation: one or more possible translations?

In the introduction, three important advantages of document translation over query translation were mentioned. Firstly, it can be done off-line. Secondly, if a machine translation system is used, it is possible to present the user a high quality preview of a document. Thirdly, there is more context available for explicit lexical disambiguation which might lead to better retrieval performance in terms of precision and recall. For several types of applications, the first and second advantage may be a good reason to choose for document translation. The third advantage seems quite plausible and was hypothesised in a number of early publications on cross-language retrieval, e.g. by Oard and Dorr (1996), Hull and Grefenstette (1996) and Kraaij (1997).

Does the document translation approach to cross-language retrieval using classical machine translation *really* lead to better retrieval performance than the query translation approach using a machine readable dictionary? A recent experimental study by Oard (1998) suggests it does. However, for a number of reasons it is very difficult to answer this question on the basis of empirical

evidence. A first problem is that in the query translation approach, searching is done in the language of the documents while in the document translation approach searching is done in the language of the query. But information retrieval is probably not equally difficult for each language, for instance because some languages (e.g. Finnish) have a much more complex morphology than other languages (e.g. English). A second problem is that, for a sound answer to the question, it is necessary to have a machine translation system and a machine readable dictionary that have exactly the same lexical coverage. If the machine translation system misses vital translations that the machine readable dictionary does list, one ends up comparing the coverage of the respective translation lexicons instead of the two approaches to cross-language retrieval. Within the Twenty-One project there is a third, more practical, problem that prevents evaluation of the usefulness of the used translation system (LOGOS) against the usefulness of the machine readable dictionaries available within the project (Van Dale). The Van Dale dictionaries are entirely based on Dutch head words, but translation from and to Dutch is not supported by LOGOS. These considerations urge us to rephrase the issue into a more manageable question.

A first, manageable, step in comparing document translation with query translation might be the following. What is, given a translation lexicon, the best approach for query translation: using one translation for each query term (i.e. explicit disambiguation) or using all possible translations? Picking one translation is a necessary condition for the document translation approach. For query translation one can either use one translation for searching, or more than one. The choice for either one or more translations also reflects the classical precision / recall dilemma in information retrieval: picking one specific translation of each query term is a good strategy to achieve high precision; using all possible translations of each query term is a good strategy to achieve high recall.

6.3   Methods for query translation

As stated in the previous section, one of the issues dealt with in this chapter is comparing cross-language information retrieval using one translation per query term with retrieval using more than one translation per query term. Results will be reported of retrieval experiments using the Dutch queries on the English TREC cross-language task collection. A Dutch query will be referred to as the source language query; the English query will be referred to as the translated query. The experiments can be divided into three categories:

1. query translation using one translation per source language query term;

2. query translation using unstructured queries of all possible translations per source language query term;

3. query translation using structured queries of all possible translations per source language query term.

6.3.1 Using one translation per query term

If only one translation per query term is used for searching, the translation process must have some kind of explicit disambiguation procedure. This procedure might be based on an existing machine translation system, or alternatively, on statistical techniques or heuristics. After disambiguation, the translated query can be treated the way a query is normally treated in a monolingual setting. A 'normal' monolingual setting in this context might be retrieval on the basis of one of the ranked retrieval models presented in section 2.3. Of course, the basic model of section 4.2 will be used instead. For the sake of completeness, the ranking algorithm is repeated below.

$$\text{score}_{\text{C1}}(d) \;=\; \log(\textstyle\sum_t tf(t,d)) + \sum_{i=1}^{n} \log(1 + \frac{\lambda\, tf(t_i,d)(\sum_t df(t))}{(1-\lambda)\, df(t_i)(\sum_t tf(t,d))})$$

Figure 6.1 gives an example of an English request "third, world" that is used to search a French collection. Although both "third" and "world" might have more than one possible translation, the system has to pick one of them.
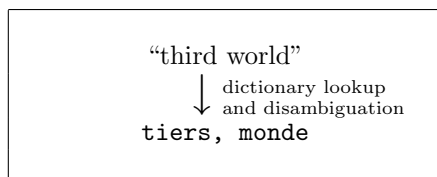
Figure 6.1: Using one translation per query term

In section 6.4 a number of heuristics and statistics for disambiguation will be explored. As explained in section 6.2 it is not possible to actually use machine translation for disambiguation. It is however possible to define an upper bound on what is achievable with the one-translation approach by asking a human expert to manually disambiguate the output of the machine readable dictionary. It is hypothesised that query translation using a machine translation system with the same lexical coverage as the machine readable dictionary will not result in better retrieval performance than query translation using the manually disambiguated output of the same dictionary.

6.3.2 Using unstructured queries

If more than one translation per source language query term is used for searching we might still treat the translated query as an unstructured bag-of-words. As will be shown in section 6.5, the way of weighting the possible translations is crucial for unstructured queries. In particular it is important to normalise the possible translations in such a way that for each source language query term the weights of possible translations sum up to one. Not using normalisation

will make source language query terms with a lot of possible translations un-intentionally more important than source language query terms that have less possible translations. For the unstructured query runs statistical translation was added 'artificially' by making the number of times a query term occurs in equation 4.2 proportional to the translation probabilities. The ranking algorithm used is the following, where Q is the bag-of-words containing the possible translations q of all source query terms, and $w(q)$ is the weight of a possible translation q. Note that the algorithm uses the new model in a way that was not originally intended: the number of times a query term occurs in the query is replaced by the translation probabilities / weights. A similar generalisation of the query term frequency is used by Ng (2000).

$$\text{score}_{c2}(d) \; = \; \log(\textstyle\sum_t tf(t,d)) + \sum_{q \in Q} w(q) \cdot \log(1 + \frac{\lambda \, tf(q,d)(\sum_t df(t))}{(1-\lambda) df(q)(\sum_t tf(t,d))})$$

Figure 6.2 again gives the example of an English query (`third`, `world`) that is used to search a French collection. It is assumed that the English term `third` has two possible French translations: `tiers` and `troisième` and that the English term `world` has three possible translations: `monde`, `mondial` and `terre`. Instead of selecting one translation we might use all possible translations to search the document collection. The results of the translation module in figure 6.2 could be
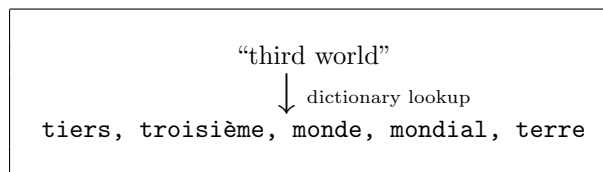
Figure 6.2: Translation using an unstructured query

used directly for searching the French collection (see `run2a` in section 6.5), but this would make the term `world` in the source language query more important (because it has more possible translations) than the word `third`. Normalisation of the possible translations might therefore be used to make the contribution of `third` as high as the contribution of `world`. In this case the possible translations of `third` are reweighted to 0.5 and the possible translations of `world` to 0.33 (see `run2c` in section 6.5). If one of the possible translations of a source language query term is more probable than the other(s), this possible translation might be weighted higher than the other(s) while keeping the normalisation in tact (`run2d` in section 6.5).

6.3.3   Using structured queries

Treating all possible translations as one unstructured bag-of-words, ignores the fact that a document containing one possible translation of each source language

query term is more likely to be relevant than a document containing all possible translations of only one source language query term. Structuring the queries as suggested in section 4.3 should show better results than unstructured queries. For the experiments with structured queries, a variant of equation 4.16 with document frequencies and document length correction was used, which is the formula shown below. In the formula, $\tau_i(j)$ is the probability of the jth possible translation of ith source language query term, which is by definition zero for all possible translations that were not suggested by the translation module.

$$\text{score}_{\text{C3}}(d) \ =$$
$$\log(\textstyle\sum_t tf(t,d)) + \sum_{i=1}^{n} \log(1 + \frac{\lambda\left(\sum_{j=1}^{m} \tau_i(j) tf(t^{(j)},d)\right) \sum_t df(t)}{(1-\lambda)(\sum_{j=1}^{m} \tau_i(j) df(t^{(j)})) \sum_t tf(t,d)})$$

Figure 6.3 again gives the example of an English query (`third world`) on a French document collection by using the representation of structured queries introduced in section 4.3. The structured query reflects the possible translations of the source language query terms in an intuitive way.
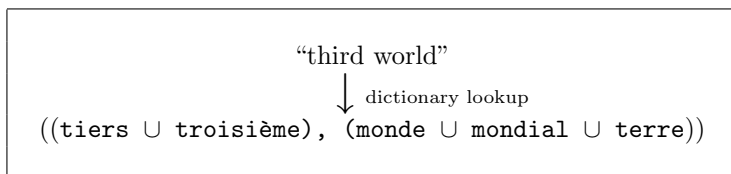
Figure 6.3: Translation using a structured query

6.4 Heuristics and statistics for disambiguation

This section lists a number of information resources that can be used to identify the correct translation or translations of a query term. The section briefly describes information that is explicitly or implicitly in the dictionary and information from other sources like parallel corpora and the document collection itself.

6.4.1 Dictionary preferred translation

The VLIS lexical database of Van Dale Lexicography lists for each entry explicitly one *preferred* translation which is considered the most commonly used one. Replacing each query term with the preferred translation is a simple, but possibly effective, approach to cross-language retrieval.

6.4.2 Pseudo frequencies

The Van Dale database also contains explicit information on the sense of possible translations. Some Dutch head words carry over to the same English transla-

tion for different senses. For example the Dutch head word "jeugd" may be translated to "youth" in three senses: the sense of 'characteristic', 'time-frame' and 'persons'. The 'persons' sense has a synonym translation: "youngsters". As "youth" occurs in the dictionary under three senses and "youngsters" under one sense, we assign "youth" a weight that is three times as high as the weight for "youngsters". The assumption made by weighting translations is that the number of occurrences in the dictionary may serve as rough estimates of actual frequencies in parallel corpora. In other words: the number of occurrences in the dictionary serve as *pseudo frequencies*. Ideally, if the domain is limited and parallel corpora on the domain are available, weights should be estimated from actual data as described in the next section.

6.4.3   Frequencies from parallel corpora

The Twenty-One system contains documents on the domain of sustainable development. Translation in Twenty-One is done using a general purpose dictionary (Van Dale) and a general purpose MT-system (LOGOS), but these resources are not very well suited for domain-specific jargon. Domain-specific jargon and its translations are implicitly available in parallel corpora on sustainable development. Translation pairs can be derived from parallel corpora using statistical co-occurrence by so-called word alignment algorithms. Within the Twenty-One project word alignment algorithms were developed that do the job in a fast and reliable way (Hiemstra, De Jong, and Kraaij 1997; Hiemstra 1998b). Domain specific translation lexicons were derived from Agenda 21, a UN-document on sustainable development that is available in most of the European languages including Dutch and English.

For the experiment, the automatically derived dictionary was merged with the Van Dale dictionary in the following way. For each entry, the pseudo frequencies and the real frequencies of the possible translations were added. Pseudo frequencies are usually not higher than four or five, but the real frequencies in the parallel corpus may be more than a thousand for frequent translation pairs. Adding pseudo frequencies and real frequencies has the effect that for possible translations that are frequent in the corpus the real frequencies will be important, but for translations that are infrequent or missing the pseudo frequencies will be important.

Translation pairs that have a frequency of one or two in the parallel corpus may-be erroneously derived by the word alignment algorithm. If, however, such an infrequent translation pair is also listed in the machine readable dictionary, then the pair was probably correct. Therefore we added a bonus frequency of three to each possible translation that is both in the corpus and in Van Dale.

6.4.4   Context for disambiguation

The techniques introduced so far do not resemble techniques that are actually used in machine translation systems. Traditionally, disambiguation in machine translation systems is based on (syntactic) context of words. In this section

a statistical algorithm is introduced that tries to translate the request words in context. The algorithm uses candidate noun phrases extracted from the document base to disambiguate the query. Noun phrases were extracted using the standard tools as used in the Twenty-One system: the Xerox morphological tools and the TNO parser. The noun phrases were sorted and then counted, resulting in a list of unique phrases with frequency of occurrence.

The introduction of noun phrases (or any multi-word expression) in the translation process leads to two types of ambiguity: sense ambiguity and structural ambiguity. Figure 6.4 gives an example of the French translation chart

-		
tiers monde	guerre mondiale	
troisième tiers	monde mondiale terre	guerre bataille
third	world	war

Figure 6.4: Translation chart of "third world war"

of the English noun phrase "third world war". Each word in this noun phrase can have several translations, which are displayed in the bottom cells of the chart, the so-called sense ambiguity. According to a list of French noun phrases there may be two candidate multi-word translations: `tiers monde` for the English noun phrase "third world" and `guerre mondiale` for "world war". These candidate translations are displayed in the upper cells of the chart. Because the internal structure of noun phrases was not available for the translation process, a full noun phrase can be translated by decomposing it in several ways. For example "third world war" can be split up in the separate translation of either "third world" and "war" or in the separate translation of "third" and "world war". The most probable decomposition can be found using techniques developed for stochastic grammars (Bod 1995). The probabilities of the parse trees can be mapped into probabilities, or weights, of possible translations. A more detailed description of the algorithm can be found in (Kraaij and Hiemstra 1998).

6.4.5 Manual disambiguation

The manual disambiguation of the dictionary output was done by a qualified interpreter who also was a native speaker of English. She had access to the Dutch version of the topics and to the English dictionary output consisting of a number of possible translations per source language (Dutch) query word. For each Dutch word, one of the possible English translations had to be chosen, even if the correct translation was not one of them.

6.4.6   Other information

In the experiments described in this chapter we ignored one important source of information: the multi-word entries in the Van Dale dictionaries. Multi-word expressions like for instance "world war" are explicitly listed in the dictionary. For the experiments described in this chapter we only used word-by-word translations using the single word entries. Multi-word entries might be used in future evaluations in combination with the extensions for proximity searching introduced in section 4.8.

6.5   Experimental setup and results

In section 6.3 we identified three methods for query translation: using one translation per query term, using an unstructured query of all translations per source language query term, and using a structured query of all translations per source language query term. Each method is assigned a number 1, 2 or 3. In section 6.4 five sources of information were identified that may be used by these methods: dictionary preference, pseudo frequencies, parallel corpora, context in noun phrases and human expertise. Given the five information sources we identified seven (two unstructured query experiments were done both with and without normalisation) retrieval experiments or "runs" which are listed in table 6.1. Each experiment is labelled with a letter from *a* to *g*. The combinations

run name	technique to weight translations / pick the best translation
run ?a	no weights used / dictionary preferred translation.
run ?b	weight by pseudo frequencies.
run ?c	normalise weights of possible translations (run?a)
run ?d	weight by normalised pseudo frequencies
run ?e	normalised 'real' frequencies estimated from the parallel Agenda 21 corpus.
run ?f	weight by using noun phrases from documents (including normalisation)
run ?g	disambiguation by a human expert

Table 6.1: Disambiguation methods

of seven information sources and three methods define a total number of 21 possible experiments. After removing combinations that are redundant or not informative 15 experiments remain.

In the remainder of this section we will report the results of 15 experiments on the TREC cross-language task test collection (Braschler et al. 1999) topics 1-24. The Dutch topics were used to search the English documents. Experiments were compared by the average precision over ranks of relevant documents, average precision in short. Additionally, the result of each experiment will be compared

with the result of a monolingual base line run, which is the result of queries based on the English version of the TREC topics. The monolingual run performs at an average precision of 0.372.[1] All experiments were done with the experimental language models retrieval engine developed at the University of Twente.

6.5.1 One translation runs

Table 6.2 lists the results of the one translation runs. Normalisation of translation weights is not useful for picking the best translation. Therefore the table does not list `run1c` and `run1d`. (`run1d` would give exactly the same results as `run1b`.)
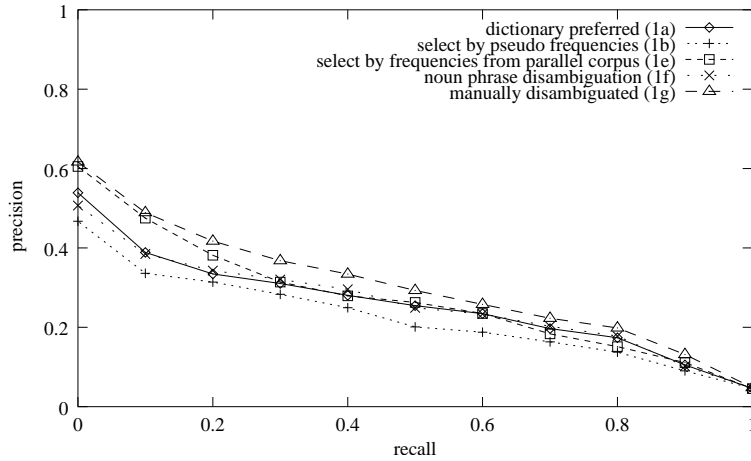
Figure 6.5: Recall-precision plots of one-translation experiments

Not surprisingly, the manually disambiguated run outperforms the automatic runs. Translation ambiguity and missing terminology are the two primary sources of cross-language retrieval error (Hull and Grefenstette 1996), so it is hypothesised that the loss of performance is due to missing terminology and possibly errors in the translation scripts. If the average precision measure is used for comparison between two runs, then the manually disambiguated run performs at 78 % of the monolingual base line. This might be seen as an upper bound on what is possible using a one-translation approach on the TREC cross-language collection. Average precision results are listed in table 6.2

The best automatic run is the run using corpus frequencies `run1e`. This is a surprise, because a relatively small corpus was used on the domain of the Twenty-One demonstrator which is *sustainable development*. Inspection of the topics however learns us that a lot of topics discuss international problems like air pollution, combating AIDS, etc., which fall directly in the domain of sustainable development. The dictionary preferred run `run1a` performs reasonable well.

---

[1]The results differ from the results reported in (Hiemstra and De Jong 1999), which only used the 21 topics that were available at the time of TREC-6.

run	precision at document:			average precision	relative to baseline
	10	30	100		
run 1a	0.330	0.262	0.169	0.246	66 %
run 1b	0.291	0.253	0.147	0.211	57 %
run 1e	0.365	0.300	0.178	0.258	69 %
run 1f	0.348	0.286	0.175	0.247	66 %
run 1g	0.404	0.336	0.214	0.292	78 %

Table 6.2: Results of one-translation experiments

The run using context from noun phrases `run1f` performs only a little better. Pseudo frequencies `run1b` are less useful for identifying the correct translation.

6.5.2   Unstructured query runs

All unstructured query runs use the translation probabilities from the different translation and disambiguation modules, except for run `run2a` and `run2b` which use translation frequencies instead of probabilities. Table 6.5.2 lists the results of the unstructured query runs using all possible translations of each word in the request.
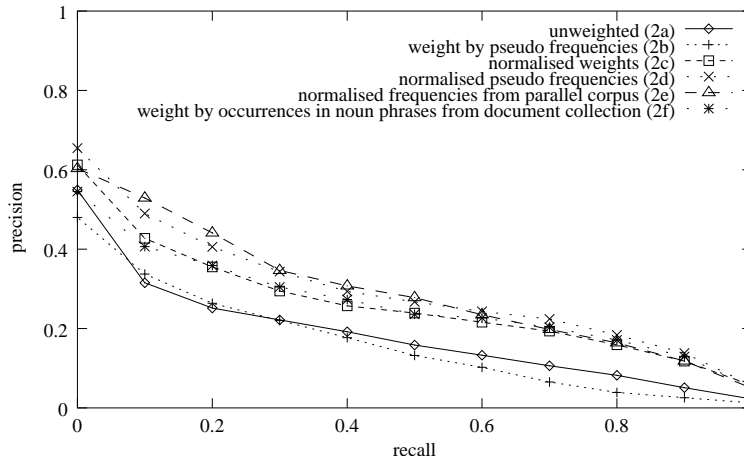
Figure 6.6: Recall-precision plots of unstructured query experiments

A first important thing to notice is that the normalisation of the term weights is a prerequisite for good performance if all possible translations per source language query term are used in an unstructured query. Not using the normalisation, as done in `run2a` and `run2b` will drop performance to a disappointing 40 to 45 % of the monolingual base line. More surprisingly, the pseudo frequency run `run2d` and the real frequency run `run2e` now perform almost equally well

and both approach the upper bound on what is possible with the one translation approach (`run1g`). Although the pseudo frequencies are not very useful for identifying the best translation, they seem to be as realistic as real frequencies if used for estimating the translation probabilities

run	precision at document:			average precision	relative to baseline
	10	30	100		
run 2a	0.300	0.240	0.167	0.169	45 %
run 2b	0.291	0.228	0.138	0.151	40 %
run 2c	0.378	0.320	0.196	0.249	67 %
run 2d	0.404	0.352	0.214	0.285	77 %
run 2e	0.426	0.354	0.213	0.281	75 %
run 2f	0.378	0.315	0.204	0.254	68 %

Table 6.3: Results of unstructured query experiments

Pairwise comparison with the one translation experiments by the use of average precision shows the following. If the identical methods are compared, that is a comparison of parallel corpus runs, noun phrase runs, etc., then none of the differences between the one-translation experiments and the unstructured queries experiment is statistically significant. Details can be found in the appendix.

6.5.3   Structured query runs

Table 6.4 lists the results of the structured query runs. Normalisation of term weights is implicit in the structured query, so `run3a` and `run3b` will give exactly the same results as `run3c` and `run3d` respectively.

The four runs do not differ as much in performance as their unstructured equivalents, which suggests that the structured queries are more robust than the unstructured queries. Again, the pseudo frequency run `run3d` and the real frequency run `run3e` perform almost equally well. Table 6.4 shows that three out of four runs perform better than the manually disambiguated one-translation run `run1g`.

Pairwise comparison with the corresponding one-translation experiments shows the following. All structured query runs outperform the corresponding one-translation runs that use identical methods for disambiguation. The differences between the pseudo frequencies experiments (`run1b` vs. `run3d`) and the parallel corpus experiments (`run1e` vs. `run3e`) are statistically significant at the 5 % level. The differences between dictionary preferred and unweighted structured queries is not significant (`run1a` vs. `run3c`). The difference between the noun phrase runs is also not significant (`run1f` vs. `run3f`).

Pairwise comparison of the structured query experiments with their corresponding unstructured query experiments shows the following. All structured
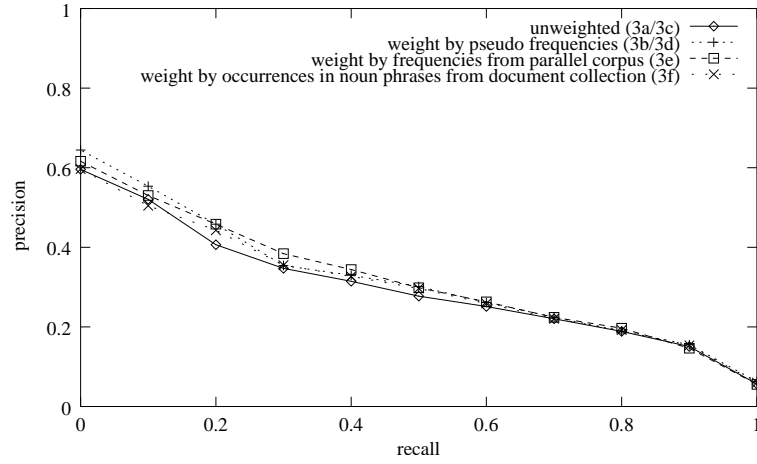
Figure 6.7: Recall-precision plots of structured query experiments

run	precision at document:			average	relative to
	10	30	100	precision	baseline
run 3c	0.396	0.354	0.226	0.289	78 %
run 3d	0.444	0.361	0.231	0.307	82 %
run 3e	0.439	0.362	0.229	0.309	83 %
run 3f	0.422	0.354	0.225	0.298	80 %

Table 6.4: Results of structured query experiments

query runs outperform their corresponding unstructured runs. The unweighted experiments (`run?c`) and the parallel corpus experiments (`run?e`) differ significantly at the 5 % level. The differences between the pseudo frequency experiments and the noun phrase experiments are not significant.

Pairwise comparison of the structured query experiments with the manually disambiguated experiment shows that none of the differences is statistically significant. Details can be found in table C.5 of the appendix.

6.5.4   Some post-hoc experiments

Some post-hoc experiments were done to see how structured queries, without the use of translation probabilities, perform if the p-norm model is used. The results of this experiment should be compared to the language model-based experiment `run 3c`, which also does not use translation probabilities. For the p-norm model, $p = 2$ and Ltu weights were used, a combination that was relatively successful in the manually formulated Boolean queries experiment. A second monolingual experiment `base 2` was done using the vector space model with Lnu.ltu weights

for fair comparison.

A second unrelated experiment `run 3r` was done that uses the relevant document to estimate relevance weights and translation probabilities retrospectively using the EM-algorithm for structured queries.

run	precision at document:			average precision	relative to baseline
	10	30	100		
base 2 (Lnu.ltu)	0.496	0.393	0.252	0.329	-
run 3p (*p*-norm Ltu $p=2$)	0.252	0.217	0.164	0.159	48 %
run 3r (LM retrosp. relw.)	0.561	0.417	0.264	0.377	101 %

Table 6.5: Results of post-hoc structured query experiments

The *p*-norm experiment shows that the *p*-norm model cannot cope with structured queries that are generated automatically by a translation module. Its performance is only 48 % of the Lnu.ltu base line which reaches an average precision of 0.329. The difference between the language model-based structured query run, that does not use translation probabilities (`run 3c`) is significant at the 1 % level.

Retrospective relevance weighting for structured queries reaches an average precision of 0.377, which outperforms the language model's monolingual base line. Note that some queries miss vital translations. If translations are better, retrospective relevance weighting might be much better than the monolingual baseline (Hiemstra, Kraaij, Pohlmann, and Westerveld 2000). As said in chapter 5, the results of retrospective relevance weighting are of theoretical interest only: the algorithm does what it was designed for.

6.5.5 Pool validation

Judgements for the cross-language task are probably not as complete as the judgements for the other TREC tasks because of the relatively small number of participants. For topics 1-24 of the cross-language collection, the judgements were done on a per-language basis. Of the 19 judged runs, only 6 were runs on the English collection that was used for the experiments in this chapter. For the official TREC-6 cross-language task runs, each run that contributed to the pool was evaluated both with (standard evaluation) and without the relevant documents that the run uniquely contributed to the pool. The difference between the two evaluations will give an idea of how reliable the collections are for future work. The comparison shows that on average, an unjudged run will have 0.022 higher average precision after judging. The standard deviation of the differences between judged and unjudged runs is more than half of the mean, indicating considerable variation among runs. Details of the pool validation experiment can be found in table C.9 of the appendix. The same experiment on the TREC-7 CLIR pool shows similar results. Note that if the systems are ordered by their

performance, the order after judging is different from the order before judging. This suggests that some caution has to be taken on the results reported in this chapter.

6.6   Discussion

The experiments in this chapter have shown that the structured query experiments consistently outperform the automatic one-translation runs and the unstructured query runs for a number of different disambiguation methods. There is some evidence that the differences are significant, for instance from the experiments that use the frequencies from the parallel corpus. Structured queries also consistently outperform the experiment that uses manual disambiguation of the translation output. None of these differences is significant, but it suggests that no automatic explicit disambiguation method could possibly outperform the method introduced in this book.

Unfortunately, differences between methods are hard to detect because of the relatively small number of topics that was used for this evaluation. Also, the collection might be less reliable than the main TREC collection because of the relatively small number of official participants. It is therefore desirable to redo this evaluation on another test collection using a reasonable amount of topics.

In several early publications on cross-language retrieval (Hull and Grefenstette 1996; Kraaij 1997; Oard and Dorr 1996) it is hypothesised that the document translation approach to cross-language retrieval leads to better retrieval performance than the query translation approach because there is more context available in documents for lexical disambiguation. Of course, lexical disambiguation is easier if there is more context available, but the results suggest that lexical disambiguation is not essential for good retrieval performance. In fact, table 6.4 shows that the best performing runs simply use all possible translations. The results of the manually disambiguated run suggest that not much can be gained by putting a lot of effort in explicit disambiguation of possible translations. By using the statistical translation model of information retrieval, disambiguation is done implicitly during searching. This suggests that the hypothesis that document translation leads to better retrieval performance than query translation might not be true after all: further research is needed on this topic.

Chapter 7

Adaptive Information Filtering

This chapter reports on the evaluation of a prototype adaptive filtering system. The main question this chapter tries to answer is whether it is possible to improve upon a base line system by using the relevance weighting algorithm of the information retrieval language model. Section 7.1 briefly introduces adaptive filtering systems. Section 7.2 introduces the prototype adaptive filtering system. Finally, section 7.3 reports of the evaluation results.

7.1 Introduction

Today, most people can be reached anytime, anywhere. We have mobile telephones, voice mail, short message service (SMS), facsimile, electronic mail, etc., etc. Some people get completely lost in information space, especially if one adds keeping up with news groups, the world wide web, newspapers, radio, and television. Some researchers suggest that overload of information may lead to psychological and physical problems, called information fatigue syndrome or information stress (Wurman 1989). As a possible solution to information stress, adaptive information filtering systems actively disseminate personalised information to the user.

7.1.1 Filtering systems

When a document is received by the filtering system, it is matched against a user profile. A user profile is the query of information filtering systems:[1] it contains information about the user's interests. Once the user has entered her profile, the filtering system can do its work. In contrast to queries that are entered in information retrieval systems, the profile remains relatively stable, whereas

---

[1]At any place where this chapter mentions "profile" one might read "query" and vice versa.

the collection is dynamic. Documents come in one at a time or in relatively
small batches. If a received document matches the profile with a score that is
higher than a certain threshold, the user is notified. Users are able to control
the filtering process by giving feedback to the system, which uses the user's
feedback to adapt the profile and the threshold. Positive feedback will result in
more documents on the subject of the received document and negative feedback
will result in less documents, thereby minimising the chance of information
stress.

   The design of an adaptive filtering system raises three important problems
(Ekkelenkamp, Kraaij, and Van Leeuwen 1999): firstly, how to set the initial
threshold, secondly how to adapt the threshold and thirdly, how to adapt the
user profile. Setting the thresholds probably has the greater impact on per-
ceived performance (Zhai et al. 1999). Once the threshold algorithms perform
satisfactory, it is hard to improve upon the performance by query reweighting.
Although this chapter reports on some work on the development of the threshold
algorithms, the main objective of this chapter is the evaluation of the relevance
weighting algorithm introduced in section 4.4.2. It is hypothesised that a base
line filtering system, that uses the language model matching algorithm and an
adaptive thresholding algorithm, can be improved significantly by the use of the
relevance weighting algorithm.

7.1.2   The utility of a filtering system

Filtering systems either show an incoming document to the user, or not. The
output of a filtering system is not a ranked list of documents, but simply a set of
unordered documents. In principle, the system could therefore be evaluated by
using the simple set-based definitions of precision and recall without the need to
average them over e.g. document cut-offs. However, precision and recall do not
capture the fact that the filtering system should keep the number of selected
documents as small as possible. Compare for instance a filtering system that
shows the user one non-relevant document a week, with a filtering system that
shows the user 100 non-relevant documents a week. If both systems are not able
to select any relevant documents, for instance because there were no relevant
documents that week, they both get zero precision. Surely the former system
does a much better job than the latter system (Hull 1999). As an alternative,
the following two utility measures will be used.

$$\mathrm{LF}_1 \;=\; 3\,r \,-\, 2\,(n{-}r) \qquad r : \text{number of relevant documents selected}$$
$$\phantom{\mathrm{LF}_1 \;=\; 3\,r \,-\, 2\,(n{-}r) \qquad} n : \text{number of documents selected}$$
$$\mathrm{LF}_2 \;=\; 3\,r \,-\, (n{-}r) \qquad\;\; R : \text{total number of relevant documents}$$

   The utility measures LF_1 and LF_2 assign a value or cost to each document,
based on whether it is relevant or not. The first measure represents a user
for which a relevant selected document has a value of 3, and a non-relevant
selected document has a cost of 2. This user needs to see at least 2 relevant
documents in each 5 selected. If not, the utility will become negative and we

might imagine the user suffering from information stress. The second measure represents a user whose costs of reading a non-relevant document are twice as low. Two versions of the prototype system will be tested, one optimised for LF_1 and one optimised for LF_2. The systems will be evaluated by the measures for which they are optimised. The higher the utility score of a system for a user profile, the better the system is performing. The two-tailed sign test will be used to test if the differences between methods are significant at the 5 % level (see appendix A).

The remainder of this chapter is organised as follows. Section 7.2 describes the prototype adaptive filtering system. The evaluation method and the system's results are described in section 7.3.

7.2 A prototype adaptive filtering system

A prototype adaptive filtering system was built using the experimental language models retrieval engine which was also used for the cross-language experiments. The evaluation was part of the official TREC-8 adaptive filtering task.

7.2.1 The background corpus

As adaptive filtering systems receive the documents in a sequence, global information on terms, like document frequencies or collection frequencies, will not be available to the system. Initial document frequencies for term weighting are therefore collected from a background corpus. The background corpus is a set of documents that is available during system development. If possible, the background corpus should contain documents that are similar to the documents stream that is to be filtered. By using the document frequencies of the background corpus, the system can use the language model's retrieval algorithms as introduced in section 4. Whenever a document is received, the document frequencies of the terms that occur in the document are increased by one. This way, the document frequencies will more and more reflect the document stream, and zero document frequencies are avoided.

7.2.2 Setting the initial threshold

The new model's retrieval algorithm assigns to each incoming document the probability that the document's language model generates the user profile. For ranking this is sufficient, but for binary selection of a document the question "when is the probability high enough?" needs to be answered. One way to answer this question is to relate the probability of sampling the profile from a document to the probability that the profile is the result of a random sample from the entire collection. Profiles that have a high probability of being sampled from the collection (i.e. profiles with common words), should receive a higher initial threshold than profiles with a low probability of being sampled from the collection (i.e. profiles with uncommon words). The probability that a profile

$T_1, T_2, \cdots, T_n$ of length n is sampled from the collection might be defined as follows.

$$P(T_1 = t_1, T_2 = t_2, \cdots, T_n = t_n) = \prod_{i=1}^{n} \frac{df(t_i)}{\sum_t df(t)} \qquad (7.1)$$

Initially only documents that generate the profile with a much higher probability than equation 7.1 should be selected. The initial threshold might be set to select documents with probabilities that are more than 100,000 times higher than the probability of random selection. This value was found empirically on a different collection. A value of 100,000 does not result in a very high threshold, because words that appear only once in a large background corpus will receive a very small probability compared to the probability of a matching term.

After rewriting the probability measures to their corresponding vector product weighting algorithms (see section 4.7), the document frequencies in the initial threshold disappear. The vector product threshold that corresponds with the decision above is $threshold = n \log(1 \, / \, (1 - \lambda_i)) + c$, where $c = \log(100,000)$. This shows an interesting feature of the initial threshold. In its vector product form, the threshold is related to the importance weights λ_i. High initial importance weights result in a high initial threshold. Importance weights are initialised to $\lambda_i = 0.5$ and re-estimated after feedback from the user.

7.2.3   Threshold adaptation

The threshold adaptation algorithm is the part of the system that uses the utility functions to optimise its performance. It was simply decided to aim just below the optimum utility given the scores of the documents that were selected by the system. Updating was done as follows.

1. Recompute the scores of all selected documents (because of changed document frequencies and changed relevance weights);
2. recompute the initial threshold (because of changed relevance weights λ_i) and add it to the selected documents as if it were a non-relevant document;
3. rank the selected documents by their new scores and find the maximum utility max by walking down the ranked list;
4. the new threshold will be the score of the lowest ranked document that has a utility of $max - 3$ when optimising for LF_1 and $max - 1$ when optimising for LF_2.

As long as the system does not find any relevant document, it will increase its threshold quite fast. In general, it will never lower its threshold again, although this might happen in practice because changed document frequencies and importance weights sometimes change the ranking of selected documents.

7.2.4   Relevance weighting of query terms

Initially, when no information on relevant documents is available, each term in the profile will get the same importance weight $\lambda_i = 0.5$. So, initially we

assume that the profile is best explained if on average half of the profile terms is sampled from relevant documents and the other half is sampled from the updated background corpus frequencies. If a relevant document is available, it might be possible to explain the profile better. The EM-algorithm for re-estimation of importance weights λ_i will make sure that terms that occur often in the relevant documents that are selected so far, get a high importance weight λ_i. Profile terms that do not occur (often) in the relevant documents are more likely to be sampled from the updated background corpus frequencies and get a low importance weight λ_i.

7.3 Experimental results

This section describes a controlled experiment that emulates the stream of incoming documents and the user's feedback.

7.3.1 Evaluation setup

The 1992, 1993 and 1994 editions of the Financial Times were used to emulate a three year long document stream. The collection contains 204,790 documents, which corresponds to about 187 documents a day. The Financial Times is a subset of the TREC collection, which makes it possible to use topics 351-400 to build profiles that represent the profiles of 50 users. The prototype adaptive filtering system now processes the collection in chronological order. Each document is matched against the profiles of the 50 hypothetical users. If the score exceeds the threshold of a profile, the document is sent to the user who will give feedback to the system: either relevant or non-relevant. The feedback is emulated by using the TREC relevance judgements. The system is only allowed to use judgements of documents that were sent to the user, emulating the fact that users can only give feedback on documents that were sent to them. The prototype system processes the three year document stream in about 20 hours, so one system could in theory process over 1,000 times as many documents for 50 users in real time.

For the background corpus, the '87 to '91 editions of the Wall Street Journal were used. Later editions of the Wall Street Journal were not used because this data would not have been available in a real world application. The topics and The Financial Times documents were stemmed using the Porter stemmer and stopped using the Smart stop-list which was augmented with some domain-specific stop words like "document" and "relevant". The topic's title, narrative and description were used to build the initial profile. The controlled language fields of the Financial Times test collection were not used. We did not process the incoming documents in chunks. That is, document frequencies were updated for each incoming document; a binary decision was made directly for each incoming document; selected documents were immediately checked for relevance; thresholds and profiles were immediately updated after the relevance assessments. Unjudged documents were assumed to be not relevant. All selected

documents were saved for future updating of thresholds and query profiles.

7.3.2 Results

Six different strategies were applied: three optimised for LF_1 and three optimised for LF_2. For both utility functions the same three experiments were done.

1. A baseline run that only uses the initial threshold setting and threshold adaptation routines;
2. the same run as 1, but with relevance weighting of profile terms;
3. the same run as 1, but using a very high initial threshold.

The high initial threshold experiments were done to check whether a very conservative threshold algorithm could possibly be more beneficial than a query reweighting technique. These two experiments were done using the TNO retrieval engine under slightly different conditions. The TNO system used the AP News wire data as the background corpus to estimate the initial document frequencies from and a somewhat different stop list. The slightly different conditions did not change the big picture of the evaluation results. The two runs will not be used for pair-wise comparisons.

run	LF_1	LF_2	prec.	recall
LF_1 optimised	-9.30	4.86	0.242	0.240
LF_1 optimised; profile reweighting	-7.28	7.10	0.243	0.251
LF_1 optimised; high initial threshold	-1.20	2.46	0.216	0.105
LF_2 optimised	-12.96	4.80	0.232	0.254
LF_2 optimised; profile reweighting	-9.12	6.60	0.237	0.254
LF_2 optimised; high initial threshold	-5.54	1.34	0.199	0.127

Table 7.1: Adaptive filtering results averaged over topics

Table 7.1 lists the evaluation results of the runs using four evaluation measures: LF_1, LF_2, precision and recall averaged over topics. The utility scores reported are averaged over the 50 test profiles. Precision and recall were averaged over the profiles by assigning 0 % recall to topics with no relevant documents and assigning 0 % precision to topics with empty retrieved sets. Despite the potential problems with the precision and recall measures, they are reported as well because precision and recall serve easy comparison with the experiments reported in previous chapters. Also, precision and especially recall contain valuable information about the size of the selected sets, which is not explicitly provided by the utility measures.

Both baseline runs show a consistent improvement in the average utility, average precision and average recall after relevance weighting of query terms. The improvements are not significant according to the sign test. Interestingly, query reweighting has a different impact on the two systems. It causes improved recall for the LF_1 system and improved precision for the LF_2 system: The

LF_1 system selected 5 % more documents after query reweighting, but the LF_2 system selected 8 % less documents.

The high initial threshold runs show different behaviour. When optimising for LF_1 (run 1p), the performance in terms of average utility improves considerably. At the same time, the performance in terms of precision and recall goes down. When optimising for LF_2, a high initial threshold results in a system with lower performance than the baseline in terms of average utility, precision and recall. The high initial threshold experiments reveal a problem with the LF_1 utility. For this measure, it is plain too hard to build a system that does not perform below zero utility on average. Scoring negatively on utility means that the user would prefer to use no system at all. Interestingly, the high initial threshold for the LF_1 system did not select any document for 22 out of 50 topics. It improved its utility at the cost of precision and recall and by doing so it came pretty close to no system at all.

7.4 Discussion

A comparison between the systems with and without relevance weighting using the sign test does not detect a significant difference on the utility score. The Differences in both precision and recall are also not significant. This does not mean that the methods are not different, but just that the sign test was not able to detect it. For both utility measures, the relevance weighting algorithm improved the performance of the base line system in terms of utility, precision and recall. A very conservative threshold algorithm is only beneficial in terms of utility, if the system performs below the level that is acceptable for the user. In future experiments, further improvements might be possible from structured initial profiles that are built by expanding each term with synonyms and other related variants. In this case relevance reweighting might be able to select the right variant from each group of synonyms.

Chapter 8

Conclusions

This chapter concludes this book by answering the three research questions introduced in chapter 1 and by summarising the evaluation results. Section 8.1 summarises the contributions of this thesis to information retrieval theory by summarising the application of statistical language models to term weighting, relevance feedback and structured queries. Section 8.2 summarises the model's implications for the formulation of structured queries from natural language search statements. Section 8.3 reports on the results of a practical comparison of a language model-based system with systems based on some well-established models, and on the evaluation of two prototype retrieval systems.

8.1 Contributions to IR theory

This section summarises the answers to the first research question that was introduced in chapter 1: *How to apply the theory of statistical language models to three classical problems of information retrieval modelling: term weighting, relevance feedback and structured queries?*

8.1.1 The basic model and term weighting

A basic model of information retrieval has been introduced that defines the matching process of retrieval systems. For each document in the collection, a language model defines the typical language use that belongs to that document. The probability that the document's language model generated the user query, is used to rank the documents. The probability mechanism that generates the query, explicitly distinguishes important terms and unimportant terms. The probability of an important term is defined by the probability of drawing the term at random from the document. The probability of an unimportant term is defined by the probability of drawing the term at random from the collection. Assuming that the query terms are independent, there is a term weighting algorithm that assigns zero weight to non-matching terms.

Conclusion 1 A *tf.idf* term weighting algorithm can be derived from a formal model of information retrieval.

The algorithm uses *tf.idf* weighting if document frequencies are used to specify the probabilities of the unimportant terms. Note that the *derivation* of a term weighting algorithm from a theory is stronger than the *motivation* of a term weighting algorithm from a theory. That is, if a term weighting algorithm is derived from a formal model then it can be motivated by that model, but motivation does not imply derivation. The *tf.idf* term weighting algorithms have for instance been motivated by the 2-Poisson model (Robertson and Walker 1994) and by information theoretic measures (Aizawa 2000), but by following these motivations many *tf.idf* measures seem reasonable, including the algorithms that perform badly in experimental settings.

8.1.2   Importance of query terms and relevance feedback

The probability mechanism that defines how a query is generated from a document leaves one unknown parameter for each query term. The parameter is associated with the binary event "importance of a query term".

Conclusion 2 The probability of the importance of a query term is a value of the usefulness of a term to retrieve relevant documents.

The probability of the importance of a query term is called the 'importance of a term', or the 'importance weight' in this book. The weight does not depend directly on the number of occurrences of the term in the collection, and can be used to model the following four seemingly unrelated issues in information retrieval: stop words, mandatory terms, coordination level rankings, and relevance weighting of query terms.

Conclusion 3 Traditional removal of a stop term from the query can be modelled by assuming that the importance of the term is 0.

For the well-established models of information retrieval, stop word removal is something that is specified outside the model, not within the model itself.

Conclusion 4 A term that is mandatory in the retrieved documents, can be modelled by assuming that the importance of the term is 1.

If the importance of a term is 1, then all documents that do not contain the term will be assigned zero probability. Modelling of mandatory terms is not possible in the well-established models.

Conclusion 5 If the average importance of the terms in a query is close to 1, then the system obeys the conditions of coordination level retrieval.

Coordination level ranking is a partial ranking in which documents that contain k query terms are always ranked above documents containing $k-1$ query terms. User studies have shown that users prefer coordination level rankings over rankings that do not obey the conditions of coordination level ranking, especially if

short queries are used. Ranking by coordination level gives the user easy insight in why a certain document is ranked above another.

Conclusion 6 A relevance weighting algorithm has been developed that estimates new values for the importance weights from examples of relevant documents.

The presented EM-algorithm provides maximum likelihood estimates for the importance weights that optimise the models of the relevant documents. The algorithm optimises the joint probability of the query and each known relevant document, assuming independence between the relevant documents.

   This thesis did not suggest a query expansion method, like e.g. the relevance feedback method described in section 2.3.2, for the language model-based system. There is however nothing that prevents the development of such an algorithm. Query expansion methods were developed for language model-based systems by Miller et al. (1999) and Ng (2000).

8.1.3   The extended model and structured queries

A second model of information retrieval has been introduced that is an extension of the basic model of the matching process. The extended model adds a statistical translation step to the basic model. The statistical translation step can be looked upon as a model of the query formulation process. In practice, the system uses the translation model and the matching model as two separate steps. If a request is entered, the system first uses the translation model to hypothesise for each word in the request the terms that might have generated it. This results in a structured query that represents all queries that might have generated the request. In the second step, the system uses the basic model for each document to calculate the probability that the document generated any of the queries represented by the structured query.

Conclusion 7 The extended retrieval model provides a way to process structured queries in conjunctive normal form, where the original query (the request) forms the conjunctive query and the possible translations of each term form the disjunctive parts.

Conclusion 8 For each indexing strategy that unambiguously converts words in documents to index terms, there is a corresponding query formulation strategy for structured queries that produces the exact same retrieval results, if collection frequencies are used for the probabilities of unimportant terms.

Unambiguously means that each word is converted deterministically to one, and only one term as e.g. done by converting words to lower case or as done by a stemmer. So, there is a strategy to morphological generation that generates structured queries that produce the exact same results as the use of a stemmer during indexing and automatic query formulation.

Conclusion 9 A relevance weighting algorithm has been developed for structured queries that estimates new values for the importance weights and the translation probabilities from examples of relevant documents.

The presented relevance weighting algorithm optimises the models of the relevant documents by providing maximum likelihood estimates of the importance weights and the translation probabilities.

8.1.4   Hidden Markov models and Bayesian networks

Much of the theory underlying the information retrieval models, like for instance the use of the EM-algorithm to estimate the unknown parameters, was developed for hidden Markov models and Bayesian networks. This thesis briefly presents the language model-based system using the hidden Markov model formalism and using the Bayesian network formalism.

Conclusion 10 The basic model and the extended model can be presented as hidden Markov models.

Conclusion 11 The basic model and the extended model can be presented as Bayesian networks.

Miller, Leek, and Schwartz (1999) introduced a language model-based retrieval system using hidden Markov models. This thesis added a hidden Markov presentation for the extended retrieval model. More interesting from a historic point of view is the presentation as a Bayesian network. Bayesian networks for information retrieval were introduced by Turtle and Croft (1991). The two approaches share the fact that they infer the probability of the query from the hypothesis that the document is relevant. The language model-based Bayesian network has the following advantages over the traditional Bayesian networks for information retrieval. It does not use approximate link matrices, it does not need an additional term weighting algorithm, and it provides a way to train the model from examples of relevant documents.

8.2   Automatic query formulation

This section summarises the answers to the second research question: *How to apply the theory of statistical language models to the automatic formulation of structured queries from natural language search statements?* To master the automatic formulation of structured queries, it should be clear how to model advanced free text search facilities intended for manual query formulation.

8.2.1   Advanced search facilities for free text

Some of the solutions to advanced search facilities for manual query formulation were already introduced in the previous section. Conclusion 4 introduced a way to model mandatory terms. Conclusion 8 implies that wildcards can be modelled

by treating the terms that match the wildcarded term as possible translations of this term. Similarly, a synonym operator can be modelled by treating the terms it relates as possible translations of an unknown term. Boolean-structured queries can the processed if they are automatically converted to the conjunctive normal form.

Conclusion 12 Dependency operators can be modelled by introducing three levels of importance, a level for unimportant terms, a level for the importance of the single terms, and a level for the importance of the dependency relation.

Dependency relations are used to combine information from different sources. For instance, additional information may be provided by the position information in the index, or by the field information in the index. If the index contains position information, then terms can be related by these positions. A proximity operator uses the relative position of terms in documents to specify a search for two related words. Bigram probabilities can be used for the adjacency of terms (Miller, Leek, and Schwartz 1999; Song and Croft 1999). Similar probabilities can be used for other proximity operators. Terms might also depend on a certain record field. A record field or document field is a part of the document that can be searched separately, like for instance the title or the abstract.

8.2.2 Natural language processing

Natural language processing technology has always played an important role in information retrieval. Usually natural language processing modules are used for query formulation *and* indexing. This thesis shows how these modules can be used during query formulation only, leading to much more flexible information retrieval systems. Stemming is probably the most popular example of the application of natural language processing technology to information retrieval. According to conclusion 8, a stemmed query on a stemmed index results in the exact same ranking of documents as a structured query using the stem's morphological variants on an index that is not stemmed. This mathematical interpretation of stemming can be applied to many other linguistical tools that analyse natural language text at the lexical level.

Conclusion 13 The statistical translation model can be used to integrate modules that analyse natural language at the lexical level.

There are a lot of examples of modules that analyse natural language at the lexical level, many of which are already integrated in commercial systems. For instance, edit distance (Baeza-Yates 1992) can be used to recover from type errors or errors from optical character recognition; fuzzy matching algorithms (De Heer 1979) or the soundex algorithm (Gadd 1988) can be used to recover from spelling errors; morphological analysis can be used to match morphological variants; ontologies as Wordnet (Miller et al. 1990) can be used to match synonyms and related terms; translation can be used for cross-language retrieval, etc. Degrees of matches from these modules, e.g. from the fuzzy matching algorithm, can be used to hypothesise a translation probability.

Conclusion 14 Dependency operators can be used to integrate modules that analyse natural language at the syntactic level.

The dependency operators discussed in this thesis are proximity operators and field search operators. Proximity operators provide a basic tool for the search of phrases in documents. For instance, a noun phrase grammar might be used to find phrases in the request which are used to search for documents that contain the single terms adjacent or near to each other.

8.3   Evaluation results

This section summarises the answers to the third research question: *What is the performance of the language model-based approach compared to the performance of well-established approaches?* The language model-based algorithms were compared to well-established algorithms on three standard tasks: ad-hoc retrieval, retrospective relevance weighting and manually formulated Boolean-structured queries. Two prototype retrieval systems were developed, a prototype cross-language retrieval system and a prototype adaptive filtering system. The prototype systems evaluate respectively the language model's approach to automatic query formulation and the model's approach to relevance weighting.

8.3.1   Retrieval performance on standard tasks

The language model-based algorithms were compared with some well-established retrieval algorithms in a number of controlled experiments. Retrieval algorithms were compared on three tasks: ad-hoc retrieval, retrospective relevance weighting and manually formulated Boolean-structured queries.

Conclusion 15 On the ad-hoc task, the language model-based system outperforms four systems that use well-established retrieval algorithms. All differences are significant, except for the difference with the BM25 algorithm for which there is insufficient evidence to disprove the null hypothesis that both algorithms perform equally.

Conclusion 16 On the retrospective relevance weighting task, the language model-based system outperforms the traditional probabilistic model and the BM25 algorithm. The difference with the probabilistic model is significant. There is insufficient evidence to disprove equal performance of the language model-based system and the BM25 system.

Experimental results show that all three systems degrade the performance of some queries. This seems rather alarming, considering that training data and test data are the same for this task. The effect seems to be more severe for the BM25 algorithm, of which the difference between the ad-hoc and relevance weighting experiments is not significant at the 5 % level according to the sign test. The system based on the traditional probabilistic model and the language

model-based system do show significant improvement after relevance weighting of query terms.

It has been noted for other applications of language models, for instance for part-of-speech tagging (Elworthy 1994), that EM re-estimation sometimes degrades performance if training data and test data are the same. The problem might be related to the the maximum likelihood criterion that underlies the EM-algorithm (Jelinek 1997, page 72). The maximum likelihood criterion is not directly related to the aim of maximising the probability of relevance and so it does not necessarily lead to it. In practice, the performance gain is as good as the performance gain of the traditional model and better than the BM25 algorithm.

Conclusion 17 On the manually formulated Boolean-structured query task, the language model-based system outperforms two versions of the p-norm model. The differences with both versions are significant at the 5 % level.

Not much can be gained with the special treatment of Boolean operators. Both the language model-based system and the p-norm system using Ltu weights do not show significant improvement of structured queries when compared to unstructured versions of the same queries.

8.3.2   Cross-language information retrieval

Three approaches to cross-language retrieval were compared in this task: explicit disambiguation of the translation output, the use of unstructured queries, and the use of structured queries. For each of the three approaches, different disambiguation and translation strategies were tried.

Conclusion 18 Given a resource for automatic translation, structured queries outperform explicit disambiguation methods and unstructured queries. There is some evidence that the differences between structured queries and explicit disambiguation, and the differences between structured queries and unstructured queries are statistically significant.

If translation probabilities are estimated from parallel corpora, then the difference between structured queries and explicit disambiguation and the difference between structured queries and unstructured queries is statistically significant. Interestingly, queries that were explicitly disambiguated by a human translator do not outperform the structured queries. This suggests that explicit disambiguation, as done by e.g. machine translation systems, is not necessarily a sensible approach to cross-language information retrieval.

8.3.3   Adaptive information filtering

Two adaptive filtering systems were compared in this experiment. One system only uses a threshold adaptation algorithm. The other system uses the same threshold adaptation algorithm, but also uses the relevance feedback algorithm

for the re-estimation of importance weights. The two systems were compared on two different tasks.

Conclusion 19 Relevance weighting of query terms improves the average utility of the prototype adaptive filtering system on both tasks. There is however insufficient evidence to disprove equal performance of the two systems.

The experiment shows that examples of relevant documents can be used to predict the importance of query terms in future relevant documents.

8.4   Discussion and recommendations for future research

The language model-based approach to information retrieval presented in this thesis provides a complete theory of information retrieval, covering topics like ranking, structured queries, relevance feedback, stop words, mandatory terms, coordination level ranking, $tf.idf$ weighting, stemming, translation, extended Boolean searching, proximity searching, and record field searching. There are four reasons to prefer the language model-based approach over one of the well-established models of ranked information retrieval. Firstly, the language model-based approach provides methods for term weighting, relevance feedback and structured queries, whereas none of the existing models covers both relevance feedback and structured queries. Secondly, the language model-based approach models the matching process and the query formulation process, whereas the existing models only define the matching process. Thirdly, the language model-based approach provides ways to define proximity and field search, which are not provided by the well-established models. The fourth reason for preferring the language model-based approach is that the language model-based system performs as well as, or better than, the well-established retrieval models and algorithms in controlled experiments. Future research should emphasise the application of the language model-based theory, and search for the best ways to apply the language-model based theory to various information retrieval problems.

8.4.1   Development of a query language

The experiments reported in this thesis were done without a well-defined query language. Instead, query structure, importance weights and translation probabilities were encoded by a number of different ad-hoc schemes. For the development of a serious prototype system and for future experiments, an extensible query language is essential (see e.g. Broglio, Callan, and Croft 1994). Such a query language would define a syntax in which highly structured queries can be easily expressed, thereby facilitating easy integration of natural language processing modules and easy experimentation.

8.4.2 Experimentation

This thesis reported on two systematic evaluations in which different values of parameters were tried to find the values for which the system performs best. The first systematic exploration was done on the Cranfield collection, to find the best system setting and the best value of the average importance of query terms λ for the ad-hoc task. The second exploration was done on the TREC cross-language collection, to find the best query translation method for cross-language retrieval. These two experiments should be repeated respectively on a collection that is much bigger than the Cranfield collection, and on a collection for which more queries are available.

Similar experiments should be done for other tasks as well. The best value of the average importance of query terms λ that was experimentally found in the Cranfield collection for the ad-hoc task, was used in the manually formulated Boolean-structured query task and in the cross-language retrieval task. On these tasks, the values might be suboptimal.

Many more interesting experiments are suggested in this thesis. For instance the application of proximity operators, the combination of information from different record fields, and the application of any of the natural language processing modules mentioned above.

8.4.3 Linguistically motivated document representations

The algorithms presented in this thesis can be used on simple indexes for free text retrieval. An indexing strategy that is sufficient would identify words in free text and put them, possibly with position and field information, in the index. This raises the following question: Will the theory presented in this thesis be useful if more complex document representations are available? Examples of more complex document representations are head-modifier pairs (Strzalkowski 1995; Kraaij and Pohlmann 1998), index expressions (Bruza and Van der Weide 1992), and representations based on structured ontologies (Van Bakel 1998). These document representations are linguistically motivated and can be derived by part-of-speech taggers and parsers. Conclusion 14, which was motivated by the use of position information, suggests that the the use of linguistically motivated document representations might be an interesting way to continue the research into the use of language models for information retrieval.

Appendix A

Evaluation methodology

This appendix explains the main assumptions and background of the applied evaluation methodology. Section A.1 introduces three main ingredients of a meaningful information retrieval experiment: a test collection, a measure of the effectiveness of the search and a test to determine statistical significance between methods. The remaining sections address each of the three ingredients.

A.1 Introduction

Evaluation of a retrieval system is concerned with how well the system is satisfying users, not just in individual cases, but collectively, for all actual and potential users in the community (Tague-Sutcliffe 1996). Although some aspects of retrieval systems can be evaluated without consulting the user, ultimately some actual or potential users have to be subjects in a controlled information retrieval experiment. Doing an evaluation involving real people is not only a costly job, it is also difficult to control and therefore hard to replicate. For this reason, methods have been developed to design unbiased test collections. These test collections are created by consulting potential users, but once they are created they can be used to evaluate information retrieval systems without the need to consult the users during further evaluations. If a test collection is available, a new retrieval method can be evaluated by comparing it to some well-established methods in a controlled experiment. Hull (1993) mentions the following three ingredients of a controlled information retrieval experiment.

1. A information retrieval test collection, consisting of documents, requests and relevance judgements.

2. One or more suitable evaluation measures that assign values to the effectiveness of the search.

3. A statistical methodology that determines whether the observed differences in performance between the methods investigated are statistically significant.

Test collections consist of a large number of documents, a number of requests, and relevance judgements ("the right answers"). Test collections, and the assumptions underlying relevance are described in section A.2. The effectiveness of the search is usually measured by the combination of *precision* and *recall*. Precision is defined by the fraction of the retrieved documents that is actually relevant. Recall is defined by the fraction of the relevant documents that is actually retrieved.

$$\text{precision} \;\; = \;\; \frac{r}{n} \qquad r : \text{number of relevant documents retrieved}$$
$$\qquad\qquad\qquad\qquad n : \text{number of documents retrieved}$$
$$\text{recall} \;\; = \;\; \frac{r}{R} \qquad R : \text{total number of relevant documents}$$

For the evaluation of ranked retrieval system, precision and recall have to be averaged somehow over the ranked lists. Section A.3 describes three of these approaches. Section A.4 addresses three significance tests, each with its advantages and disadvantages. Finally, section A.5 concludes this chapter by summarising the followed procedure.

A.2   Test collections

Information retrieval test collections consist of three distinct parts: the documents, the requests and the relevance judgements or "the right answers". Today's standard test collections are constructed in the Text Retrieval Conferences, TREC in the following. When constructing a test collection, relevance judgements are the most difficult to control. Users that participate in the evaluation should be carefully instructed on how to do the judgements. Also, the documents that are to be judged should be carefully selected, because it is impossible to judge all documents if the collection is very large.

A.2.1   TREC

The TREC collections are designed by the United States National Institute of Standards and Technology. The TREC collections that are used in this book consist of newspaper and newswire data. For a standard TREC evaluation, usually 50 requests are used, which are called "topics" in TREC. Figure A.1 shows a sample topic (Voorhees and Harman 2000).

A.2.2   Assumptions about relevance

In chapter 1, the relevance of a document is defined by its usefulness for satisfying the user's information need according to the user's subjective opinion. That being said, it is good to realise that while relevance is a key notion in information science, it is also the subject of ever-lasting debates and controversies (Saracevic 1975; Mizzaro 1997). There are many aspects to relevance that are problematic for the evaluation of retrieval systems. To name a few, relevance of a document may be:

```
<num> Number:  409
<title> legal, Pan Am, 103

<desc> Description:
What legal actions have resulted from the destruction
of Pan Am Flight 103 over Lockerbie, Scotland, on
December 21, 1988?

<narr> Narrative:
Documents describing any charges, claims, or fines
presented to or imposed by any court or tribunal are
relevant, but documents that discuss charges made in
diplomatic jousting are not relevant.
```

Figure A.1: An example TREC topic

judged on a scale a document might for instance be not useful, somewhat useful, fairly useful, very useful and totally useful to a user;

dependent on time a document that is useful to the user today, might no longer be useful to the user later on;

dependent on other retrieved documents a user that walks down a ranked list might for instance find a document further down the list not useful, because it covers the exact same information as the top ranked document, whereas it would have been useful if the top ranked document was not retrieved;

multifaceted the usefulness of a document might be determined by topicality, credibility, specificity, exhaustiveness, accuracy, recency, clarity, etc.

For the test collection that are used in the evaluations in this book it is assumed that relevance is a dichotomous decision, that does not depend on other retrieved documents. The judges that did the relevance assessments were instructed to do their judgements based on these assumptions. They had to make a binary decision on each document, even if they were in great doubt and they did not let information from other documents influence their decisions. Similar assumptions about relevance are made by some retrieval models, for instance the language models presented in chapter 4, and by the probabilistic model and the inference network models presented in chapter 2.

A.2.3   The document judgements pool

To measure the recall of a system (see section A.3 on evaluation measures below), the total number of relevant documents for a topic has to be known. Ideally, subjects in TREC should therefore read and judge every document in the collection for a topic. Unfortunately this is humanly speaking impossible,

since for example the TREC-8 main collection consists of over half a million documents. As a result, test collections have to be constructed by judging only a sample of the documents for each topic. The sample is constructed for each topic as follows. Of each participating system, the top 100 documents retrieved determine the pool. Duplicate documents are removed from the pool and the remaining documents are sorted randomly, e.g. by their document identifiers. The resulting list is judged by the TREC subjects for relevance. The subjects do not know which documents were retrieved by which system, nor do they know whether a document has a high or a low ranking in one or more of the runs.

The pooling method will inevitably miss some of the relevant documents. So, in practice, TREC evaluations will only determine an upper bound on recall. Recent studies have shown that although additional searches might reveal additional relevant documents this is not likely to change the relative performance of the systems compared to each other (Buckley and Voorhees 2000). For the cross-language retrieval experiments similar tests were done for the pool of the TREC cross-language collections, to see how reliable the evaluation results are.

A.3   Evaluation measures

If relevance is binary valued then retrieval performance is usually measured by the combination of *precision* and *recall*. If the retrieval system makes a binary decision as well, that is, if the system either retrieves documents or not, without ranking them, then precision and recall are measured by fixed proportions. The overall system performance is determined by averaging precision and recall over a sufficiently large number of requests.

If the system ranks the documents in decreasing order of some document score, then the precision and recall measures should somehow be averaged over the number of documents retrieved. Several average precision and average recall measures have been suggested that model the behaviour of a user walking down a ranked list of documents. The idea is to give a number of evaluation measures for different types of users. At one end of the spectrum is the user that is satisfied with any relevant document, for instance a user that searches a web page on last nights football results. At the other end of the spectrum is the user that is only satisfied with most or all of the relevant documents, for instance a lawyer searching for jurisprudence. In TREC three different evaluation measures are used: precision at fixed levels of recall, precision at fixed points in the ranked list and the average precision over the ranks of relevant documents.

A.3.1   Precision at fixed recall levels

For this evaluation a number of fixed recall levels are chosen, for instance 10 levels: $\{0.1, 0.2, \cdots, 1.0\}$. The levels correspond to users that are satisfied if they find respectively 10 %, 20 %,$\cdots$, 100 % of the relevant documents. For each of these levels the corresponding precision is determined by averaging the precision

on that level over the topics. The resulting precision points are often visualised in a recall-precision graph. Figure A.2 shows an example. The graph shows the typical behaviour of information retrieval systems. Increasing the recall of a search implies decreasing the precision of the search. Or, by walking down a ranked list in search for more relevant documents, the chance to encounter nonrelevant documents will grow faster than the chance to encounter of relevant documents.
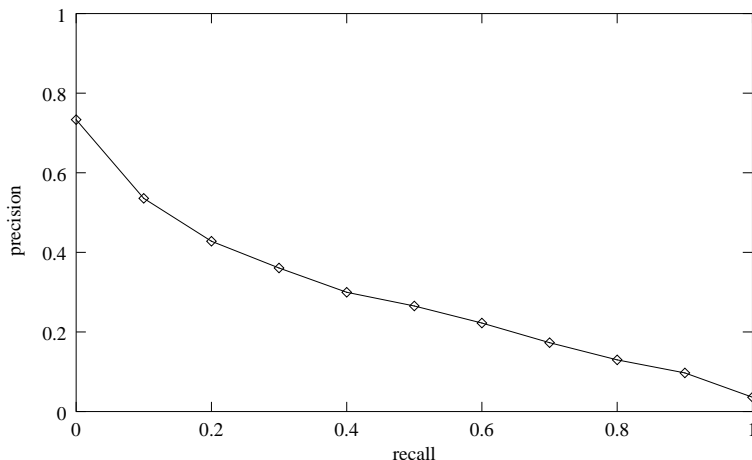
Figure A.2: Example recall-precision graph

In practice, the levels of recall might not correspond with natural recall levels. For instance, if the total number of relevant documents R is 3, then the natural recall levels are 0.33, 0.67 and 1.0. Other recall levels are determined by using interpolation. A simple but often used interpolation method determines the precision at recall level l by the maximum precision at all points larger than l. For example, if the three relevant documents were retrieved at rank 4, 9 and 20, then the precision at recall points $0.0, \cdots, 0.3$ is 0.25, at 0.4, 0.50 and 0.6 the precision is 0.22 and at $0.7, \cdots, 1.0$ the precision is 0.15 (Harman 1995). Interpolation might also be used to determine the precision at recall 0.0, resulting in a total of 11 recall levels. Sometimes one average measure, the so-called 11 points interpolated average precision, is calculate by averaging the average precision values over the 11 recall points.

A.3.2 Precision at fixed points in the ranked list

Recall is not necessarily a good measure of user equivalence. For instance if one query has 20 relevant documents while another has 200. A recall of 50 % would be a reasonable goal in the first case, but unmanageable for most users in the second case (Hull 1993). A more user oriented method would simply choose a number of fixed points in the ranked list, for instance 9 points at: 5, 10, 15, 20, 30, 100, 200, 500 and 1000 documents retrieved. These points correspond with

users that are willing to read 5, 10, 15, etc. documents of a search. For each
of these points in the ranked list, the precision is determined by averaging the
precision on that level over the topics. Similarly, the average recall might be
computed for each the points. A potential problem with these measures however
is that, although precision and recall theoretically range between 0 and 1, they
are often restricted to a small fraction of the range for many cut-off points. For
instance, if the total number of relevant documents $R = 3$, then the precision at
10 will be 0.33 at maximum. One point of special interest from this perspective
is the precision at R documents retrieved. At this point the average precision
and average recall do range between 0 and 1. Furthermore, precision and recall
are by definition equal at this point. The R-precision value is the precision at
each (different) R averaged over the topics (Harman 1995).

A.3.3   Average precision over ranks of relevant documents

The average precision measure is a single value that is determined for each topic
and then averaged over the topics. The measure corresponds with a user that
walks down a ranked list of documents that will only stop after he / she has
found a certain number of relevant documents. The measure is the average of the
precision calculated at the rank of each relevant document retrieved. Relevant
documents that are not retrieved are assigned a precision value of zero. For the
example above where the three relevant documents are retrieved at ranks 4, 9
and 20, the average precision would be computed as $(0.25 + 0.22 + 0.15)/3 =$
0.21. This measure has the advantages that it does not need the interpolation
method and that it uses the full range between 0 and 1 (Harman 1995).

A.4   Significance tests

Simply citing percentages improvements of one method over another is helpful,
but it does not tell if the improvements were in fact due to differences of the
two methods. Instead, differences between two methods might simply be due
to random variation in the performance, that is, the difference might occur by
chance even if the two methods perform equally well. To make significance
testing of the differences applicable, a reasonable amount of queries is needed.
When evaluation measures are averaged over a number of queries, one can obtain
an estimate of the error associated with the measure. (Hull 1993).

   Significance tests are designed to disprove the null hypothesis H_0. For re-
trieval experiments, the null hypothesis will be that there is no difference be-
tween method A and method B. The idea is to show that, given the data,
the null hypothesis is indefensible, because it leads to an implausible low prob-
ability. Rejecting H_0 implies accepting the alternative hypothesis H_1. The
alternative hypothesis for the retrieval experiments will be that either method
A consistently outperforms method B, or method B consistently outperforms
method A.

   A test statistic is a function of the data. It should have the following two

properties. Firstly, it should behave differently under H_0 than under H_1. Secondly, it should be possible to calculate its probability distribution under H_0. For information retrieval, there is usually much more variation in the performance per query than in the performance per system. Therefore, the test statistics used are paired tests which are based on the performance differences between the two systems for each query. The methods assume that the performance differences consist of a mean difference μ and an error ε_i for each query i, where the errors are independent. The null hypothesis is that $\mu = 0$. The following three paired tests have been used in the Smart retrieval experiments (Salton and McGill 1983, page 171).

the paired t-test assumes that errors are normally distributed. Under H_0, the distribution is Student's t with $\#queries - 1$ degrees of freedom.

the paired Wilcoxon's signed ranks test is a non-parametric test that assumes that errors come from a continuous distribution that is symmetric around 0. The statistic uses the ranks of the absolute differences instead of the differences themselves.

the paired sign test is a non-parametric test only uses the sign of the differences between method A and B for each query. The test statistic is the number of times that the least frequent sign occurs. It assumes equal probability of positive and negative errors. Under H_0, the distribution is binomial.

So, in order to use the t-test the errors must be normally distributed, and in order to use Wilcoxon's test the errors have to be continuous. However, precision and recall are discrete and bounded and therefore neither normally distributed nor continuous. The average of a reasonable number of discrete measures, like the average precision measure presented in section A.3.3, might behave similar to continuous measures and approximate the normal distribution quite well. Before the tests can be applied, the researcher has to make a qualitative judgement of the data, to check if indeed the normality assumption is reasonable (Hull 1993). If not, the sign test can be used as an alternative. For the experiments in this book, the sign test was used without checking the conditions for the other tests, following Van Rijsbergen (1979) who argues that only the sign test can be considered valid for information retrieval experiments.

A.5 Conclusion

The evaluation procedure used in chapter 5 uses the TREC test collections and follows the evaluation methods used in TREC. The three methods to average precision and recall over the ranked list of documents are all used in this book. As in TREC, the principle measure to compare two methods is the average precision over the ranks of relevant documents as presented in section A.3.3. The sign-test is used for the pair-wise comparison of two different approaches.

Appendix B

Coordination level ranking

Coordination level ranking is a partial ranking of the documents such that documents that match k query terms are always ranked above documents that match $k - 1$ query terms. Assuming that each term has the same importance weight λ, assuming a uniform document prior (equation 4.3) and using the presence weighting algorithms, the requirement is the following:

$$k \log \left(1 + m \frac{\lambda}{1 - \lambda} \right) > (k - 1) \log \left(1 + n \frac{\lambda}{1 - \lambda} \right)$$

The left-hand side of the inequality is the matching score of a document that contains k query terms. The right-hand side of the inequality is the matching score of a document that contains $k - 1$ query terms. In the inequality, m and n ($n, m > 0$) are $tf.idf$ weights of the matching terms as introduced in section 4.7.1. For the simplicity of the proof, m is taken as the minumum of the $tf.idf$ values of the k matching terms of the document on the left-hand side, and n is taken as the maximum of the $tf.idf$ values of the $k - 1$ matching terms of the document on the right-hand side. Proofing coordination level ranking for the extreme values of m and n will proof coordination level ranking for practical cases in which the $tf.idf$ values differ per matching term. Coordination level ranking might not be fulfilled if $n \gg m$.

For $k = 1$, the right-hand side is zero, and the inequality is true if $\lambda > 0$, no matter what the values of m and n are. If $k > 1$, the k's might be moved to the left-hand side of the inequality and the rest to the right-hand side, resulting in:

$$\frac{k}{k - 1} > \frac{\log \left(1 + n \frac{\lambda}{1 - \lambda} \right)}{\log \left(1 + m \frac{\lambda}{1 - \lambda} \right)}$$

The right-hand side of the inequality will go to 1 if λ approaches 1. So in the limiting case, the inequality will be true, because $k \, / \, (k-1) > 1$ for any bounded

$k > 1$. Now we only have to show that for any fixed m and n:

$$\lim_{\lambda \to 1} \frac{\log\left(1 + n\frac{\lambda}{1-\lambda}\right)}{\log\left(1 + m\frac{\lambda}{1-\lambda}\right)} = 1$$

This can be shown as follows.

$$\frac{\log\left(1 + n\frac{\lambda}{1-\lambda}\right)}{\log\left(1 + m\frac{\lambda}{1-\lambda}\right)} = \frac{\log\left(\frac{1-\lambda+n\lambda}{1-\lambda}\right)}{\log\left(\frac{1-\lambda+m\lambda}{1-\lambda}\right)} = \frac{\log(1-\lambda+n\lambda) - \log(1-\lambda)}{\log(1-\lambda+m\lambda) - \log(1-\lambda)}$$

Divding the numerator and the denominator by $-\log(1-\lambda)$ results in:

$$= \frac{1 - \dfrac{\log(1-\lambda+n\lambda)}{\log(1-\lambda)}}{1 - \dfrac{\log(1-\lambda+m\lambda)}{\log(1-\lambda)}}$$

which will in fact approach 1 if λ approaches 1, because $\lim_{\lambda \to 1} \log(1-\lambda+n\lambda) = \log n$, $\lim_{\lambda \to 1} \log(1 - \lambda + m\lambda) = \log m$, and $\lim_{\lambda \to 1} \log(1 - \lambda) = \infty$.

Appendix C

Raw evaluation results

run 1 : original *tf.idf* with cosine normalisation (tfc.tfc)
run 2 : traditional probabilistic model
run 3 : traditional probabilistic model, retrospective relevance weighting
run 4 : Lnu.ltu weighting
run 5 : BM25 weighting
run 6 : BM25, retrospective relevance weighting
run 7 : language model, version 4
run 8 : language model, version 4, retrospective relevance weighting
$+$: significant at 5 % level
$++$: significant at 1 % level

Sign test on average precision values:	
run 1 vs. run 2	-
run 1 vs. run 4	$++$
run 2 vs. run 5	$++$
run 4 vs. run 5	$++$
run 4 vs. run 7	$++$
run 5 vs. run 7	-
run 3 vs. run 6	$++$
run 3 vs. run 8	$++$
run 6 vs. run 8	-
run 2 vs. run 3	$++$
run 5 vs. run 6	-
run 7 vs. run 8	$++$

Table C.1: Significance tests of ad-hoc and rel. weighting TREC topics 301-350

run 1 : original *tf.idf* with cosine normalisation (tfc.tfc)
run 2 : traditional probabilistic model
run 3 : traditional probabilistic model, retrospective relevance weighting
run 4 : Lnu.ltu weighting
run 5 : BM25 weighting
run 6 : BM25, retrospective relevance weighting
run 7 : language model, version 4
run 8 : language model, version 4, retrospective relevance weighting

run:	1	2	3	4	5	6	7	8
Precision averages at recall values:								
0.0	0.4996	0.4563	0.4872	0.7637	0.7658	0.8100	0.7377	0.7974
0.1	0.2819	0.3135	0.3491	0.4685	0.5314	0.5722	0.5359	0.5905
0.2	0.2114	0.2467	0.2931	0.3651	0.4149	0.4523	0.4280	0.4712
0.3	0.1771	0.2300	0.2665	0.3028	0.3456	0.3795	0.3611	0.4068
0.4	0.1418	0.2032	0.2333	0.2657	0.2912	0.3243	0.2990	0.3366
0.5	0.1001	0.1851	0.2084	0.2265	0.2566	0.2726	0.2647	0.2829
0.6	0.0736	0.1516	0.1798	0.1520	0.1943	0.2286	0.2223	0.2488
0.7	0.0421	0.1061	0.1426	0.1053	0.1519	0.1632	0.1729	0.1906
0.8	0.0258	0.0785	0.1203	0.0647	0.0902	0.1131	0.1285	0.1466
0.9	0.0159	0.0593	0.0912	0.0298	0.0508	0.0679	0.0975	0.1138
1.0	0.0057	0.0322	0.0388	0.0120	0.0172	0.0316	0.0355	0.0424
Average precision:								
	0.1260	0.1647	0.1976	0.2287	0.2612	0.2888	0.2767	0.3105
Precision averages at document cut-off values:								
5	0.2920	0.2520	0.2720	0.5200	0.5400	0.6080	0.5480	0.5800
10	0.2400	0.2480	0.2680	0.4500	0.4840	0.5260	0.4940	0.5500
15	0.2307	0.2280	0.2587	0.4147	0.4453	0.4827	0.4547	0.4973
20	0.2120	0.2070	0.2430	0.3890	0.4110	0.4430	0.4200	0.4570
30	0.1867	0.1867	0.2167	0.3447	0.3660	0.4000	0.3847	0.4100
100	0.1222	0.1532	0.1716	0.2140	0.2338	0.2576	0.2354	0.2598
200	0.0910	0.1213	0.1335	0.1538	0.1636	0.1833	0.1675	0.1835
500	0.0582	0.0799	0.0844	0.0893	0.0953	0.1059	0.0991	0.1080
1000	0.0391	0.0491	0.0538	0.0555	0.0579	0.0635	0.0607	0.0656
R-precision:								
	0.1731	0.2009	0.2464	0.2818	0.3028	0.3326	0.3182	0.3490

Table C.2: Ad-hoc and relevance weighting results on TREC topics 401-450

run 1 : p-norm model, tfc weights, $p = 1$	
run 2 : p-norm model, tfc weights, $p = 2$	
run 3 : p-norm model, Ltu weights, $p = 1$	
run 4 : p-norm model, Ltu weights, $p = 2$	
run 5 : language model, ignoring query structure	
run 6 : language model	
$+$: significant at 5 % level	
$++$: significant at 1 % level	

Sign test on average precision values:

run 1 vs. run 2	$++$
run 3 vs. run 4	-
run 5 vs. run 6	-
run 2 vs. run 4	$++$
run 2 vs. run 6	$++$
run 4 vs. run 6	-

Table C.3: Significance tests of Boolean-structured queries TREC topics 301-350

run 1 : p-norm model, tfc weights, $p = 1$
run 2 : p-norm model, tfc weights, $p = 2$
run 3 : p-norm model, Ltu weights, $p = 1$
run 4 : p-norm model, Ltu weights, $p = 2$
run 5 : language model, ignoring query structure
run 6 : language model

run:	1	2	3	4	5	6
Precision averages at recall values:						
0.0	0.4706	0.5706	0.4983	0.6550	0.6832	0.6482
0.1	0.2730	0.3231	0.3743	0.4079	0.4795	0.4867
0.2	0.1621	0.1927	0.3096	0.3164	0.4007	0.4140
0.3	0.0921	0.1279	0.2433	0.2458	0.3017	0.3348
0.4	0.0737	0.0972	0.1710	0.1920	0.2538	0.2793
0.5	0.0404	0.0573	0.1308	0.1565	0.2060	0.2322
0.6	0.0276	0.0239	0.0820	0.1155	0.1645	0.1844
0.7	0.0198	0.0165	0.0490	0.0860	0.1206	0.1367
0.8	0.0139	0.0114	0.0306	0.0614	0.0763	0.0897
0.9	0.0073	0.0068	0.0161	0.0277	0.0426	0.0635
1.0	0.0039	0.0036	0.0092	0.0230	0.0300	0.0429
Average precision:						
	0.0843	0.1020	0.1564	0.1823	0.2241	0.2435
Precision averages at document cut-off values:						
5	0.2560	0.3320	0.3280	0.3920	0.4440	0.4200
10	0.2220	0.2860	0.2760	0.3660	0.3980	0.3720
15	0.1907	0.2413	0.2560	0.3333	0.3533	0.3520
20	0.1670	0.2130	0.2520	0.2970	0.3310	0.3350
30	0.1400	0.1713	0.2233	0.2600	0.3027	0.2920
100	0.0884	0.1032	0.1470	0.1800	0.1850	0.1878
200	0.0685	0.0746	0.1129	0.1307	0.1357	0.1348
500	0.0467	0.0483	0.0704	0.0782	0.0804	0.0791
1000	0.0323	0.0332	0.0467	0.0502	0.0509	0.0503
R-precision:						
	0.1089	0.1371	0.1951	0.2260	0.2702	0.2814

Table C.4: Results of Boolean-structured queries on TREC topics 301-350

run 1a : one translation, dictionary preferred
run 1b : one translation, select by pseudo frequencies
run 1e : one translation, select by frequencies from parallel corpus
run 1f : one translation, noun phrase disambiguation
run 1g : one translation, manually disambiguated

run 2a : unstructered queries, unweighted
run 2b : unstructered queries, weight by pseudo frequencies
run 2c : unstructered queries, normalised weights
run 2d : unstructered queries, normalised pseudo frequencies
run 2e : unstructered queries, normalised frequencies from parallel corpus
run 2f : unstructered queries, normalised noun phrase occurrences in doc. collection

run 3c : structured queries, unweighted
run 3d : structured queries, weight by pseudo frequencies
run 3e : structured queries, weight by frequencies from parallel corpus
run 3f : structured queries, weight by noun phrase occurrences in doc. collection
run 3p : structured queries, p-norm model $p = 2$, Ltu weighting

$+$: significant at 5 % level
$++$: significant at 1 % level

Sign test on average precision values:

run 1a vs. run 2a	-
run 1b vs. run 2b	-
run 1a vs. run 2c	-
run 1b vs. run 2d	-
run 1e vs. run 2e	-
run 1f vs. run 2f	-
run 1a vs. run 3c	-
run 1b vs. run 3d	$+$
run 1e vs. run 3e	$+$
run 1f vs. run 3f	-
run 2c vs. run 3c	$+$
run 2d vs. run 3d	-
run 2e vs. run 3e	$+$
run 2f vs. run 3f	-
run 1g vs. run 3c	-
run 1g vs. run 3d	-
run 1g vs. run 3e	-
run 1g vs. run 3f	-
run 3p vs. run 3c	$++$

Table C.5: Significance tests of cross-language runs TREC CLIR topics 1-24

base   : monolingual run (base-line)
run 1a : dictionary preferred
run 1b : select by pseudo frequencies
run 1e : select by frequencies from parallel corpus
run 1f : noun phrase disambiguation
run 1g : manually disambiguated

run:	base	1a	1b	1e	1f	1g
Precision averages at recall values:						
0.0	0.7284	0.5388	0.4673	0.6053	0.5069	0.6165
0.1	0.5885	0.3885	0.3361	0.4751	0.3846	0.4891
0.2	0.5295	0.3343	0.3139	0.3815	0.3427	0.4171
0.3	0.4818	0.3105	0.2831	0.3140	0.3203	0.3679
0.4	0.4503	0.2804	0.2493	0.2793	0.2957	0.3341
0.5	0.3831	0.2544	0.2011	0.2625	0.2490	0.2929
0.6	0.3419	0.2345	0.1876	0.2347	0.2333	0.2579
0.7	0.2892	0.1966	0.1633	0.1835	0.2025	0.2226
0.8	0.2409	0.1736	0.1374	0.1513	0.1779	0.1985
0.9	0.1735	0.1052	0.0903	0.1111	0.1018	0.1315
1.0	0.0513	0.0462	0.0462	0.0455	0.0461	0.0480
Average precision:						
	0.3723	0.2455	0.2112	0.2583	0.2474	0.2917
Precision averages at document cut-off values:						
5	0.5217	0.3652	0.2957	0.3739	0.3565	0.4261
10	0.4913	0.3304	0.2913	0.3652	0.3478	0.4043
15	0.4667	0.3014	0.2899	0.3710	0.3304	0.4058
20	0.4630	0.2957	0.2717	0.3500	0.3130	0.3674
30	0.4348	0.2623	0.2536	0.3000	0.2855	0.3362
100	0.2896	0.1691	0.1470	0.1778	0.1748	0.2135
200	0.1837	0.1130	0.1033	0.1139	0.1170	0.1398
500	0.0914	0.0615	0.0570	0.0630	0.0648	0.0781
1000	0.0503	0.0348	0.0330	0.0388	0.0371	0.0447
R-precision:						
	0.3861	0.2632	0.2199	0.2841	0.2578	0.3213

Table C.6: Baseline and one translation experiments TREC CLIR topics 1-24

run 2a : unweighted
run 2b : weight by pseudo frequencies
run 2c : normalised weights
run 2d : normalised pseudo frequencies
run 2e : normalised frequencies from parallel corpus
run 2f : weight by occurrences in noun phrases from doc. collection

run:	2a	2b	2c	2d	2e	2f
Precision averages at recall values:						
0.0	0.5492	0.4800	0.6124	0.6547	0.6032	0.5451
0.1	0.3154	0.3373	0.4272	0.4898	0.5289	0.4064
0.2	0.2517	0.2635	0.3553	0.4058	0.4410	0.3589
0.3	0.2217	0.2205	0.2946	0.3435	0.3465	0.3047
0.4	0.1919	0.1768	0.2571	0.2935	0.3075	0.2730
0.5	0.1586	0.1321	0.2389	0.2673	0.2778	0.2362
0.6	0.1328	0.1023	0.2160	0.2433	0.2348	0.2267
0.7	0.1062	0.0653	0.1934	0.2240	0.1977	0.2031
0.8	0.0822	0.0388	0.1592	0.1838	0.1641	0.1710
0.9	0.0510	0.0257	0.1178	0.1379	0.1181	0.1292
1.0	0.0223	0.0128	0.0572	0.0569	0.0496	0.0534
Average precision:						
	0.1691	0.1507	0.2490	0.2849	0.2807	0.2537
Precision averages at document cut-off values:						
5	0.3478	0.2957	0.4435	0.4783	0.4522	0.4261
10	0.3000	0.2913	0.3783	0.4043	0.4261	0.3783
15	0.2783	0.2899	0.3623	0.3942	0.4116	0.3565
20	0.2652	0.2565	0.3609	0.3891	0.3761	0.3457
30	0.2406	0.2275	0.3203	0.3522	0.3536	0.3145
100	0.1665	0.1383	0.1961	0.2143	0.2130	0.2039
200	0.1139	0.0904	0.1309	0.1417	0.1393	0.1393
500	0.0623	0.0542	0.0677	0.0730	0.0754	0.0740
1000	0.0377	0.0337	0.0395	0.0413	0.0435	0.0413
R-precision:						
	0.1862	0.1905	0.2660	0.2982	0.3068	0.2719

Table C.7: Results of unstructured queries TREC CLIR topics 1-24

run 3c : unweighted
run 3d : weight by pseudo frequencies
run 3e : weight by frequencies from parallel corpus
run 3f : weight by noun phrase occurrences in doc. collection

base 2 : monolingual run using Lnu.ltu
run 3p : structured queries using p-norm, Ltu weights $p = 2$
run 3r : retrospective relevance weighting for structured queries

run:	3c	3d	3e	3f	base 2	3p	3r
Precision averages at recall values:							
0.0	0.5957	0.6445	0.6163	0.5966	0.7192	0.5317	0.7750
0.1	0.5203	0.5536	0.5302	0.5047	0.5983	0.3112	0.6294
0.2	0.4065	0.4581	0.4580	0.4430	0.5243	0.2608	0.5781
0.3	0.3472	0.3543	0.3841	0.3552	0.4660	0.2236	0.5101
0.4	0.3147	0.3298	0.3441	0.3268	0.3997	0.1768	0.4309
0.5	0.2773	0.3016	0.2983	0.2967	0.3108	0.1374	0.3624
0.6	0.2512	0.2614	0.2629	0.2587	0.2561	0.1180	0.3228
0.7	0.2203	0.2248	0.2240	0.2204	0.2192	0.0840	0.2789
0.8	0.1886	0.1894	0.1967	0.1921	0.1730	0.0660	0.2369
0.9	0.1507	0.1553	0.1463	0.1536	0.0952	0.0244	0.1639
1.0	0.0566	0.0628	0.0556	0.0601	0.0098	0.0055	0.0666
Average precision:							
	0.2891	0.3066	0.3094	0.2978	0.3289	0.1593	0.3772
Precision averages at document cut-off values:							
5	0.4174	0.4783	0.4522	0.4174	0.5304	0.2870	0.5826
10	0.3957	0.4435	0.4391	0.4217	0.4957	0.2522	0.5609
15	0.3913	0.4203	0.4203	0.4261	0.4638	0.2493	0.4899
20	0.3783	0.4000	0.3913	0.4022	0.4457	0.2348	0.4696
30	0.3536	0.3609	0.3623	0.3536	0.3928	0.2174	0.4174
100	0.2257	0.2313	0.2287	0.2248	0.2522	0.1635	0.2635
200	0.1530	0.1561	0.1550	0.1511	0.1693	0.1152	0.1737
500	0.0805	0.0814	0.0821	0.0780	0.0892	0.0663	0.0910
1000	0.0448	0.0455	0.0469	0.0433	0.0502	0.0388	0.0502
R-precision:							
	0.3104	0.3159	0.3198	0.3119	0.3511	0.1865	0.3987

Table C.8: Results of structured queries TREC CLIR topics 1-24

run name	average precision		difference		unique
	unjudged	judged			rel.
97lsiLEE	0.0933	0.1041	0.0108	11.6 %	48
Cor6EEsc	0.3755	0.3910	0.0155	4.1 %	62
ETHee1	0.3299	0.3669	0.0370	11.2 %	160
INQ4xl1	0.2391	0.2457	0.0066	2.8 %	40
TNOee	0.2332	0.2537	0.0205	8.8 %	83
XRCECLE2EM	0.3752	0.4172	0.0420	11.2 %	165
		max:	0.0420	11.6 %	165
		mean:	0.0221	8.3 %	93
	standard deviation:		0.0144	3.9 %	56

Table C.9: TREC-6 CLIR English pool validation

run name	average precision		difference		unique
	unjudged	judged			rel.
98EITdes	0.1919	0.1962	0.0043	2.2 %	45
98EITful	0.2514	0.2767	0.0253	10.1 %	159
98EITtit	0.1807	0.1841	0.0034	1.9 %	27
BKYCL7AG	0.2345	0.2406	0.0061	2.6 %	44
BKYCL7AI	0.2012	0.2184	0.0172	8.6 %	120
BKYCL7ME	0.3111	0.3391	0.0280	9.0 %	164
RaliDicAPf2e	0.1405	0.1687	0.0282	20.1 %	176
TW1E2EF	0.1425	0.1569	0.0144	10.1 %	107
ceat7f2	0.1808	0.2319	0.0511	28.3 %	293
ibmcl7al	0.2939	0.3168	0.0229	7.8 %	135
lanl982	0.0296	0.0487	0.0191	64.5 %	140
tno7ddp	0.2174	0.2382	0.0208	9.6 %	152
tno7edpx	0.2551	0.2846	0.0295	11.6 %	109
umdxeof	0.1448	0.1610	0.0162	11.2 %	140
		max:	0.0511	64.5 %	293
		mean:	0.0205	14.1 %	129
	standard deviation:		0.0124	16.1 %	67

Table C.10: TREC-7 CLIR pool validation (four languages)

Bibliography

Aizawa, A. (2000). The feature quantity: An information theoretic perspective of tfidf-like measures. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 104–111.

Allan, J., J. Callan, F. F. Feng, and D. Malin (2000). Inquery and TREC-8. In *Proceedings of the eighth Text Retrieval Conference TREC-8*, NIST Special Publication 500-246, pp. 637–644.

AltaVista (1996). *Main page.* http://www.altavista.com

Baeza-Yates, R. A. (1992). Introduction to data structures and algorithms related to information retrieval. In W. B. Frakes and R. A. Baeza-Yates (Eds.), *Information Retrieval: Data Structures & Algorithms*, pp. 13–27. Prentice-Hall.

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

Bakel, B. van (1998). Modern classical indexing: a linguistic contribution to knowledge-based IR. In *21st ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 333–334.

Ballesteros, L. and W. B. Croft (1998). Resolving ambiguity for cross-language retrieval. In *Proceedings of the 21st ACM SIGIR Conference Research and Development in Information Retrieval (SIGIR'98)*, pp. 64–71.

Belkin, N. J. and W. B. Croft (1987). Retrieval techniques. *Annual Review of Information Science and Technology 22*, pp. 109–145.

Bengio, Y. (1999). Markovian models for sequential data. *Neural Computing Surveys 2*, 129–162. http://www.icsi.berkeley.edu/~jagota/NCS/

Berger, A. and J. Lafferty (1999). Information retrieval as statistical translation. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, pp. 222–229.

Bod, R. (1995). *Enriching Linguistics with Statistics: Performance Models for Natural Language.* Ph.D. Thesis, Department of Linguistics, Universiteit van Amsterdam.

Bookstein, A. and D. R. Swanson (1974). Probabilistic models for automatic indexing. *Journal of the American Society for Information Science 25*(5), 313–318.

Braschler, M., J. Krause, C. Peters, and P. Schäuble (1999). Cross-language information retrieval (CLIR) track overview. In *Proceedings of the seventh Text Retrieval Conference (TREC-7)*.

Brenner, E. H. (1996). *Beyond Boolean: New Approaches to Information Retrieval*. National Federation of Abstracting and Information Services.

Broglio, J., J. P. Callan, and W. B. Croft (1994). Inquery system overview. In M. Kaufmann (Ed.), *Proceedings of the TIPSTER Text Program (Phase I)*, pp. 40–48.

Broglio, J., J. P. Callan, W. B. Croft, and D. W. Nachbar (1995). Document retrieval and routing using the Inquery system. In *Proceedings of the third Text Retrieval Conference TREC-3*, pp. 29–38.

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

Bruza, P. D. and T. van der Weide (1992). Stratified hypermedia structures for information disclosure. *The Computer Journal 35*(3), 208–220.

Buckley, C., J. Allan, and G. Salton (1994). Automatic routing and ad-hoc retrieval using Smart. In *Proceedings of the second Text Retrieval Conference TREC-2*, pp. 45–55.

Buckley, C. and E. M. Voorhees (2000). Evaluating evaluation measure stability. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 33–40.

Chowdhury, G. G. (1998). *Introduction to modern information retrieval*. John Wiley & Sons.

Church, K. W. and W. A. Gale (1999). Inverse document frequency: a measure of deviation from Poisson. In A. et al. (Ed.), *NLP using Very Large Corpora*. Kluwer Academic Publishers.

Clarke, C. L. A., G. V. Cormack, and E. A. Tudhope (1997). Relevance ranking for one to three term queries. In *Proceedings of RIAO'97*, pp. 388–400.

Croft, W. B. (1993). Knowledge-based and statistical approaches to text retrieval. *IEEE Expert 8*(2), 8–12.

Croft, W. B. and D. J. Harper (1979). Using probabilistic models of document retrieval without relevance information. *Journal of Documentation 35*(4), 285–295.

Cutting, D., J. Kupiec, J. Pedersen, and P. Sibun (1992). A practical part-of-speech tagger. In *Proceedings of Applied Natural Language Processing*, pp. 133–140.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the em-algorithm plus discussions on the paper. *Journal of the Royal Statistical Society 39*(B), 1–38.

Ekkelenkamp, R., W. Kraaij, and D. van Leeuwen (1999). TNO TREC-7 site report: SDR and filtering. In *Proceedings of the seventh Text Retrieval Conference, TREC-7*, pp. 519–526. NIST Special Publication 500-242.

Elworthy, D. (1994). Does Baum-Welch re-estimation help taggers? In *Proceedings of the 4th ACL Conference on Applied Natural Language Processing (ANLP-94)*, pp. 53–58.

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

Fuhr, N. (1992). Probabilistic models in information retrieval. *The Computer Journal 35*(3), 243–255.

Fuhr, N. (1995). Probabilistic datalog: A logic for powerful retrieval methods. In *Proceedings of the 18th ACM Conference on Research and Development in Information Retrieval (SIGIR'95)*, pp. 282–290.

Gadd, T. N. (1988). Fishing fore werds: Phonetic retrieval of written text in information retrieval systems. *Program 22*(3), 222–237.

Geerts, G. and C. A. den Boon (Eds.) (1999). *Van Dale groot woordenboek der Nederlandse taal*. Van Dale Lexicografie BV.

Gey, F. C. (1994). Inferring probability of relevance using the method of logistic regression. In *Proceedings of the 17th ACM Conference on Research and Development in Information Retrieval (SIGIR'94)*, pp. 222–231.

Greiff, W. R., W. B. Croft, and H. R. Turtle (1997). Computationally tractable probabilistic modeling of boolean operators. In *Proceedings of the 20th ACM Conference on Research and Development in Information Retrieval (SIGIR'97)*, pp. 119–128.

Harman, D. K. (1991). How effective is suffixing? *Journal of the American Society for Information Science 42*(1), 7–15.

Harman, D. K. (1992). Ranking algorithms. In W. B. Frakes and R. Baeza-Yates (Eds.), *Information Retrieval: Data Structures & Algorithms*, pp. 363–392. Prentice Hall.

Harman, D. K. (1995). Evaluation techniques and measures. In *Proceedings of the third Text Retrieval Conference TREC-3*, pp. A5–A13.

Harman, D. K., E. Fox, and Baeza-Yates (1988). Inverted files. In W. B. Frakes and R. Baeza-Yates (Eds.), *Information Retrieval: Data Structures and Algorithms*, pp. 28–43. Prentice Hall.

Harter, S. P. (1975). An algorithm for probabilistic indexing. *Journal of the American Society for Information Science 26*(4), 280–289.

Hawking, D. and P. Thistlewaite (1996). Relevance weighting using distance between term occurrences. Technical Report TR-CS-96-08, The Australian National University. http://cs.anu.edu.au/techreports/

Heckerman, D. E. (1991). *Probabilistic Similarity Networks*. MIT Press.

Heer, T. de (1979). Quasi comprehension on natural language simulated by means of information traces. *Information Processing & Management 15*, 89–98.

Hiemstra, D. (1998a). A linguistically motivated probabilistic model of information retrieval. In *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pp. 569–584.

Hiemstra, D. (1998b). Multilingual domain modeling in Twenty-One: automatic creation of a bi-directional translation lexicon from a parallel corpus. In P. A. Coppen, H. van Halteren, and L. Teunissen (Eds.), *Proceedings of eighth CLIN meeting*, pp. 41–58.

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. and F. M. G. de Jong (1998). Cross-language retrieval in Twenty-One: using one, some or all possible translations? In *Proceedings of the 14th Twente Workshop on Language Technology (TWLT-14)*, pp. 19–26.

Hiemstra, D. and F. M. G. de Jong (1999). Disambiguation strategies for cross-language information retrieval. In *Proceedings of the third European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pp. 274–293.

Hiemstra, D., F. M. G. de Jong, and W. Kraaij (1997). A domain specific lexicon acquisition tool for cross-language information retrieval. In *Proceedings of RIAO'97 Conference on Computer-Assisted Searching on the Internet*, pp. 255–266.

Hiemstra, D. and A. P. de Vries (2000). Relating the new language models of information retrieval to the traditional retrieval models. Technical Report TR-CTIT-00-09, Centre for Telematics and Information Technology. http://www.ub.utwente.nl/webdocs/ctit/1/00000022.pdf

Hiemstra, D. and W. Kraaij (1999). Twenty-One at TREC-7: Ad-hoc and cross-language track. In *Proceedings of the seventh Text Retrieval Conference TREC-7*, pp. 227–238. NIST Special Publication 500-242.

Hiemstra, D., W. Kraaij, R. Pohlmann, and T. Westerveld (2000). Twenty-one at clef: Translation resources, merging strategies and relevance feedback. In *Proceedings of the 1st Workshop on Cross-Language Information Retrieval and Evaluation (CLEF-1)*, (in press).

Hotbot (1995). *Main page.* http://www.hotbot.com

Huibers, T. W. C. (1996). *An Axiomatic Theory for Information Retrieval.* Ph.D. thesis, Department of Computer Science, Utrecht University.

Hull, D. (1993). Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th ACM Conference on Research and Development in Information Retrieval (SIGIR'93)*, pp. 329–338.

Hull, D. (1999). The TREC-7 filter track: Description and analysis. In *Proceedings of the seventh Text Retrieval Conference, TREC-7*, pp. 33–56. NIST Special Publication 500-242.

Hull, D. A. and G. Grefenstette (1996). A dictionary-based approach to multilingual information retrieval. In *Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, pp. 49–57.

Jelinek, F. (1997). *Statistical Methods for Speech Recognition*. MIT Press.

Jong, F. M. G. de, J. L. Gauvain, D. Hiemstra, and K. Netter (2000). Language-based multimedia information retrieval. In *Proceedings of RIAO 2000 Conference on Content-based multimedia information access*.

Jordan, M. I. (Ed.) (1998). *Learning in Graphical Models*. Kluwer Academic Press.

Kekäläinen, J. (1999). *The effects of query complexity, expansion and structure on retrieval performance in probabilistic text retrieval*. Ph.D. thesis, Department of Information Studies, University of Tampere.

Kowalski, G. (1997). *Information Retrieval Systems: Theory and Implementation*. Kluwer Academic Publishers.

Kraaij, W. (1997). Multilingual functionality in the Twenty-One project. In *AAAI Symposium on Cross-Language Text and Speech Retrieval*. American Association for Artificial Intelligence.

Kraaij, W. and D. Hiemstra (1998). Cross-language retrieval with the Twenty-One system. In E. Voorhees and D. Harman (Eds.), *Proceedings of the 6th Text Retrieval Conference TREC-6*, pp. 753–761. NIST Special Publication 500-240.

Kraaij, W. and R. Pohlmann (1996). Viewing stemming as recall enhancement. In *Proceedings of the 19th ACM Conference on Research and Development in Information Retrieval (SIGIR'96)*, pp. 40–48.

Kraaij, W. and R. Pohlmann (1998). Comparing the effect of syntactic vs. statistical phrase index strategies for Dutch. In *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pp. 605–617.

Kraaij, W. and R. Pohlmann (2001). *Using Language Technology for Information Retrieval*. Ph.D. thesis, Faculty of Linguistics, Utrecht University. (to appear).

Kraaij, W., R. Pohlmann, and D. Hiemstra (2000). Twenty-One at TREC-8: using language technology for information retrieval. In *Proceedings of the eighth Text Retrieval Conference, TREC-8*, pp. 285–300. NIST Special Publication 500-246.

Krause, P. J. (1998). Learning probabilistic networks. Technical report, Philips Research Laboratories. http://www.auai.org/auai-tutes.html

Lawrence, S. and C. L. Giles (1999). Accessibility of information on the web. *Nature 400*, 107–109.

Lee, J. H. (1995). Analyzing the effectiveness of extended boolean models in information retrieval. Technical Report TR95-1501, Cornell University. http://cs-tr.cs.cornell.edu/

Losada, D. E. and A. Barreiro (1999). Using a belief revision operator for document ranking in extended boolean models. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, pp. 66–73.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics 11*(1-2), 22–31.

Luhn, H. P. (1957). A statistical approach to mechanised encoding and searching of literary information. *IBM Journal of Research and Development 1*(4), 309–317.

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

Margulis, E. L. (1993). Modelling documents with multiple Poisson distributions. *Information Processing and Management 29*, 215–227.

Maron, M. E. and J. L. Kuhns (1960). On relevance, probabilistic indexing and information retrieval. *Journal of the Association for Computing Machinery 7*, 216–244.

McEliece, R. and S. M. Aji (2000). The generalized distributive law. *IEEE Transactions in Information Theory*, (in press).

McEliece, R., D. J. C. MacKay, and J. F. Cheng (1998). Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE Journal on Selected Areas in Communication 16*(2), 140–152.

Miller, D. R. H., T. Leek, and R. M. Schwartz (1999). A hidden Markov model information retrieval system. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, pp. 214–221.

Miller, G. A., R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller (1990). Introduction to wordnet: an on-line lexical database. *International Journal of Lexicography 3*(4), 235–312.

Mish F. C. et al. (Ed.) (1983). *Webster's Ninth New Collegiate Dictionary*. Merriam-Webster Inc.

Mitra, M., C. Buckley, A. Singhal, and C. Cardie (1997). An analysis of statistical and syntactic phrases. In *Proceedings of the RIAO'97*, pp. 200–216.

Mizzaro, S. (1997). Relevance: The whole story. *Journal of the American Society for Information Science 48*(9), 810–832.

Mood, A. M. and F. A. Graybill (1963). *Introduction to the Theory of Statistics, Second edition*. McGraw-Hill.

Mooers, C. N. (1950). Information retrieval viewed as temporal signaling. In *Proceedings of the International Congress of Mathematicians*, Volume 1, pp. 572–573.

Ng, K. (2000). A maximum likelihood ratio information retrieval model. In *Proceedings of the eighth Text Retrieval Conference, TREC-8*. NIST Special Publication 500-246, pp. 483–492.

Oard, D. W. (1998). A comparative study of query and document translation for cross-language information retrieval. In *Proceedings of the Third Conference of the Association for Machine Translation in the Americas (AMTA)*.

Oard, D. W. and B. J. Dorr (1996). A survey of multilingual text retrieval. Technical report UMIACS-TR-96-19, University of Maryland. http://www.ee.umd.edu/medlab/mlir/mlir.html

Paice, C. P. (1984). Soft evaluation of boolean search queries in information retrieval systems. *Information Technology: Research and Development 3*(1), 33–42.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Pirkola, A. (1998). The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. In *21st ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 55–63.

Ponte, J. M. and W. B. Croft (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st ACM Conference on Research and Development in Information Retrieval (SIGIR'98)*.

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

Rabiner, L. R. (1990). A tutorial on hidden Markov models and selected applications in speech recognition. In A. Waibel and K. F. Lee (Eds.), *Readings in speech recognition*, pp. 267–296. Morgan Kaufmann.

Rajashekar, T. B. and W. B. Croft (1995). Combining automatic and manual index representations in probabilistic retrieval. *Journal of the American Society for Information Science 46*(4), 272–283.

Rasmussen, E. M. (1999). Libraries and bibliographical systems. In R. A. Baeza-Yates and B. Ribeiro-Neto (Eds.), *Modern Information Retrieval*, pp. 397–413. Addison-Wesley.

Ribeiro, B. A. N. and R. Muntz (1996). A belief network model for ir. In *Proceedings of the 19th ACM Conference on Research and Development in Information Retrieval (SIGIR'96)*, pp. 252–260.

Rijsbergen, C. J. van (1979). *Information Retrieval, second edition*. Butterworths. http://www.dcs.gla.ac.uk/Keith/Preface.html

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

Robertson, S. E. (1977). The probability ranking principle in IR. *Journal of Documentation 33*(4), 294–304.

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

Robertson, S. E., C. J. van Rijsbergen, and M. F. Porter (1981). Probabilistic models of indexing and searching. In R. N. Oddy et al. (Eds.), *Information Retrieval Research*, pp. 35–56. Butterworths.

Robertson, S. E. and S. Walker (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th ACM Conference on Research and Development in Information Retrieval (SIGIR'94)*, pp. 232–241.

Robertson, S. E. and S. Walker (1997). On relevance weights with little relevance information. In *Proceedings of the 20th ACM Conference on Research and Development in Information Retrieval (SIGIR'97)*, pp. 16–24.

Robertson, S. E. and S. Walker (2000). Okapi/Keenbow at TREC-8. In *Proceedings of the eighth Text Retrieval Conference TREC-8*, NIST Special Publication 500-246, pp. 151–162

Robertson, S. E., S. Walker, and M. Beaulieu (1999). Okapi at TREC-7: automatic ad hoc, filtering, vlc and interactive. In *Proceedings of the seventh Text Retrieval Conference, TREC-7*, pp. 253–264. NIST Special Publication 500-242.

Rocchio, J. J. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), *The Smart Retrieval System: Experiments in Automatic Document Processing*, pp. 313–323. Prentice Hall.

Rose, D. E. and C. Stevens (1997). V-twin: A lightweight engine for interactive use. In E. M. Voorhees and D. K. Harman (Eds.), *Proceedings of the 5th Text Retrieval Conference TREC-5*, pp. 279–290. NIST Special Publication 500-238.

Sahami, M. (1999). *Using Machine Learning to Improve Information Access*. Ph.D. thesis, Department of Computer Science, Stanford University.

Salton, G. (1971). *The SMART retrieval system: Experiments in automatic document processing*. Prentice-Hall.

Salton, G. (1989). *Automatic Text Processing*. Addison-Wesley.

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

Salton, G., E. A. Fox, and H. Wu (1983). Extended boolean information retrieval. *Communications of the ACM 26*(11), 1022–1036.

Salton, G. and M. J. McGill (Eds.) (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.

Salton, G. and C. S. Yang (1973). On the specification of term values in automatic indexing. *Jounral of Documentation 29*(4), 351–372.

Saracevic, T. (1975). Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information Science 26*, 321–343.

Savino, P. and F. Sebastiani (1998). Essential bibliography on multimedia information retrieval, categorisation and filtering. In *Slides of the 2nd European Digital Libraries Conference Tutorial on Multimedia Information Retrieval*.

Schäuble, P. (1997). *Multimedia Information Retrieval: Content-Based Information Retrieval from Large Text and Audio Databases*. Kluwer Academic Publishers.

Schiettecatte, F. (1998). Document retrieval using the MPS information server (a report on the TREC-6 experiment). In *Proceedings of the 6th Text Retrieval Conference TREC-6*, pp. 477–488. NIST Special Publication 500-240.

Sebastiani, F. (1994). A probabilistic terminological logic for modelling information retrieval. In *Proceedings of the 17th ACM Conference on Research and Development in Information Retrieval (SIGIR'94)*, pp. 122–130.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal 27*, 379–423, 623–656.

Singhal, A., C. Buckley, and M. Mitra (1996). Pivoted document length normalization. In *Proceedings of the 19th ACM Conference on Research and Development in Information Retrieval (SIGIR'96)*, pp. 21–29.

Smart (1994). *ftp-site*. ftp://ftp.cs.cornell.edu/pub/smart/

Song, F. and W. B. Croft (1999). A general language model for information retrieval. In *Proceedings of Eighth International Conference on Information and Knowledge Management, CIKM'99*.

Sparck-Jones, K. (1972). A statistical interpretation of term specifity and its application in retrieval. *Journal of Documentation 28*(1), 11–20.

Sparck-Jones, K., S. Walker, and S. E. Robertson (2000). A probabilistic model of information retrieval: Development and comparative experiments (part 1 and 2). *Information Processing & Management 36*(6), 779–840.

Strzalkowski, T. (1995). Natural language information retrieval. *Information Processing & Management 31*(3), 397–417.

Tague-Sutcliffe, J. M. (1996). Some perspectives on the evaluation of information retrieval systems. *Journal of the American Society for Information Science 47*(1), 1–3.

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

Turtle, H. R. (1991). *Inference Networks for Document Retrieval*. Ph.D. thesis, Centre for Intelligent Information Retrieval, University of Massachusetts Amherst.

Turtle, H. R. and W. B. Croft (1992). A comparison of text retrieval models. *The Computer Journal 35*(3), 279–290.

Twenty-One (1998). *Demonstrator*. http://twentyone.tpd.tno.nl/21demomooi

Vickery, B. C. (1970). *Techniques of Information Retrieval*. Butterworths.

Voorhees, E. M. (2000). The TREC-8 question answering track report. In *Proceedings of the eighth Text REtrieval Conference (TREC-8)*, pp. 77–82. NIST Special Publication 500-246.

Voorhees, E. M. and D. K. Harman (2000). Overview of the eighth text retrieval conference. In *Proceedings of the eighth Text REtrieval Conference (TREC-8)*, pp. 1–24. NIST Special Publication 500-246.

Vries, A. P. de (1999). *Content and Multimedia Database Management Systems*. Ph.D. thesis, Centre for Telematics and Information Technology, University of Twente.

Vries, A. P. de and D. Hiemstra (2000). The Mirror DBMS at TREC. In *Proceedings of the eighth Text Retrieval Conference, TREC-8*, pp. 725–734. NIST Special Publication 500-246.

Wilkinson, R., J. Zobel, and R. Sacks-Davis (1996). Similarity measures for short queries. In D. K. Harman (Ed.), *Proceedings of the 4th Text Retrieval Conference TREC-4*, pp. 277–286. NIST Special Publication 500-236.

Witten, I. H., A. Moffat, and T. C. Bell (1994). *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold.

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

Wurman, R. S. (1989). *Information anxiety*. Doubleday.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control 8*, 338–353.

Zhai, C., P. Jansen, E. Stoica, N. Grot, and D. A. Evans (1999). Threshold calibration in CLARIT adaptive filtering. In *Proceedings of the seventh Text Retrieval Conference, TREC-7*, pp. 149–156. NIST Special Publication 500-242.

Index

2-Poisson model, 24, 25

adaptive filtering, 113
ADJ operator, 43
AND operator, 42

Bayesian network models, 26, 73
Boolean model, 12
Boolean operators, 42

compound splitting, 41
cosine measure, 15
cross-language retrieval, 97

Dice's measure, 15
disambiguation, 98, 103
document translation, 97, 99

eliteness of terms, 25
exclusion operator, 46
extended Boolean model, 21, 23, 93

fields, 46
filtering, 113
fuzzy set model, 21

hidden Markov models, 70

idf weighting, 30
index translation, 97
indexing, 3
inference network model, 26
information filtering, 113
Inquery system, 33

Jaccard's measure, 15

mandatory term operator, 46
matching, 4

model, 9, 10
    2-Poisson, 24, 25
    Bayesian networks, 26, 73
    Boolean, 12
    extended Boolean, 21, 23, 93
    fuzzy set, 21
    hidden Markov, 70
    inference network, 26
    p-norm, 23, 93
    probabilistic, 18, 90, 91
    vector space, 15
morphological normalisation, 39

NEAR operator, 43
NOT operator, 42

Okapi system, 34
operators
    ADJ, 43
    AND, 42
    Boolean, 42–43
    exclusion, 46
    mandatory term, 46
    NEAR, 43
    NOT, 42
    OR, 42
    phrase, 46
    proximity, 43
    synonym, 46
    wildcards, 44
OR operator, 42
overlap measure, 15

p-norm model, 23, 93
paraphrase problem, 6
phrase extraction, 40
phrase operator, 46

161

Summary

Because of the world wide web, information retrieval systems are now used by millions of untrained users all over the world. The search engines that perform the information retrieval tasks, often retrieve thousands of potentially interesting documents to a query. The documents should be ranked in decreasing order of relevance in order to be useful to the user. This book describes a mathematical model of information retrieval based on the use of statistical language models. The approach uses simple document-based unigram models to compute for each document the probability that it generates the query. This probability is used to rank the documents. The study makes the following research contributions.

- The development of a model that integrates term weighting, relevance feedback and structured queries.
- The development of a model that supports multiple representations of a request or information need by integrating a statistical translation model.
- The development of a model that supports multiple representations of a document, for instance by allowing proximity searches or searches for terms from a particular record field (e.g. a search for terms from the title).
- A mathematical interpretation of stop word removal and stemming.
- A mathematical interpretation of operators for mandatory terms, wildcards and synonyms.
- A practical comparison of a language model-based retrieval system with similar systems that are based on well-established models and term weighting algorithms in a controlled experiment.
- The application of the model to cross-language information retrieval and adaptive information filtering, and the evaluation of two prototype systems in a controlled experiment.

Experimental results on three standard tasks show that the language model-based algorithms work as well as, or better than, today's top-performing retrieval algorithms. The standard tasks investigated are ad-hoc retrieval (when there are no previously retrieved documents to guide the search), retrospective relevance weighting (find the optimum model for a given set of relevant documents), and ad-hoc retrieval using manually formulated Boolean queries. The application to cross-language retrieval and adaptive filtering shows the practical use of respectively structured queries, and relevance feedback.

Samenvatting

Door het wereldwijde web gebruiken tegenwoordig miljoenen ongetrainde gebruikers over de gehele wereld informatiezoeksystemen. De zoekmachines die de zoektaken uitvoeren leveren vaak duizenden documenten op die mogelijk interessant zijn voor de gebruiker. De documenten dienen te worden geordend op relevantie om bruikbaar te zijn voor de gebruiker. Dit boek beschrijft een wiskundig model voor het zoeken van informatie dat gebaseerd is op statistische taalmodellen. De aanpak gebruikt simpele document-gebaseerde unigrammodellen om voor elk document de kans te berekenen dat het de zoekvraag genereert. Deze kans wordt gebruikt om de documenten te ordenen. De studie levert de volgende wetenschappelijke bijdragen.

- De ontwikkeling van een model waarbinnen termweging, relevantieterugkoppeling en gestructureerde zoekvragen geïntegreerd zijn.
- De ontwikkeling van een model dat meerdere representaties van een verzoek of informatiebehoefte ondersteunt, door de integratie van een statistisch *ver*taalmodel.
- De ontwikkeling van een model dat meerdere representaties van een document ondersteunt, bijvoorbeeld het zoeken naar in elkaars nabijheid voorkomende termen of het zoeken naar termen die in een bepaald veld voorkomen (bijv. het zoeken naar termen uit de titel).
- Een wiskundige verklaring voor het verwijderen van stopwoorden en het herleiden van woorden tot de stam.
- Een wiskundige verklaring voor operatoren voor verplichte termen, jokers en synoniemen.
- Een praktische vergelijking van een op taalmodellen gebaseerd zoeksysteem met vergelijkbare systemen die gebaseerd zijn op gevestigde modellen en termwegingsalgoritmen in een gecontroleerd experiment.
- De toepassing van het model op zowel het zoeken in anderstalige informatie als het adaptief filteren van informatie, en de evaluatie van twee prototype systemen in een gecontroleerd experiment.

Experimentele resultaten op drie standaardtaken laten zien dat de algoritmen die gebaseerd zijn op taalmodellen evengoed presteren als, of beter presteren dan, de best-presterende systemen van vandaag. De onderzochte standaardtaken zijn: ad-hoc zoeken (wanneer er nog geen eerder gevonden documenten zijn om het zoeken richting te geven), terugkoppeling met terugwerkende kracht (het vinden van een optimaal model voor een gegeven verzameling van relevante documenten), en het ad-hoc zoeken met behulp van handmatig geformuleerde Booleaanse zoekvragen. De toepassing op het zoeken naar anderstalige informatie en het adaptief filteren van informatie laat het praktische gebruik van respectievelijk gestructureerde zoekvragen en relevantieterugkoppeling zien.