Adaptive and Temporally-dependent Document Filtering

een wetenschappelijke proeve
op het gebied van de
Natuurwetenschappen, Wiskunde en Informatica

Proefschrift

ter verkrijging van de graad van doctor aan
de Katholieke Universiteit Nijmegen,
volgens besluit van het College van Decanen
in het openbaar te verdedigen op

woensdag 20 Juni 2001
des voormiddags om 11.00 uur precies

door

Avi Arampatzis

geboren op 1 maart 1973
te Xanthi, Griekenland

Promotor: Prof. C.H.A. Koster

Co-promotor: Dr. ir. Th.P. van der Weide

Manuscriptcommissie: Prof. D. Christodoulakis
University of Patras, Greece

Prof. S.E. Robertson
Microsoft Research Ltd, Cambridge, UK, and
City University, London, UK

Dr. T. Strzalkowski
State University of New York at Albany, USA

ISBN 90-9014891-4

στους γονείς μου,
με αγάπη και ευγνωμοσύνη

to my parents,
with love and gratitude

Contents

Foreword

This is the place to express thanks to friends and colleagues who were persuaded or bribed to read and criticize early drafts of the manuscript. I thank Petros Draggiotis for going error-hunting in the first draft, and Marco Devillers for helping me out with the Dutch summary in the last pages of this thesis. I am grateful to Arjen van de Vries, Djoerd Hiemstra, Stephen Robertson, and Tomek Strzalkowski for commenting on the research issues. My collaboration with André van Hameren has been very enjoyable. It is also a pleasure to acknowledge helpful interactions with Jean Beney and Charles Peters. I am indebted to Rachael Rafter for the extensive proofreading, for her quick reactions during difficult moments, and for designing the picture on the cover of this book. Throughout the last few years, I have enjoyed all brainstorming with Kees Koster, and I have appreciated and learned from the organized way of working with my supervisor Theo van der Weide.

Avi Arampatzis
May 6, 2001

Chapter 1

Introduction

The digital and networking revolution over the last decade has made large amounts of digital information available. This tremendous increase in digital information has led to a new challenge in *information seeking* (Oard and Marchionini, 1996). Currently, users every day find themselves confronted with large amounts of information in the form of news, e-mail messages, and especially World-Wide Web pages. Although users have access to a rich body of information, only a small fraction of this is actually relevant to the interests of any particular user. In order to reduce the effort a user has to put into determining which information is relevant to his or her interests, an automated solution is indispensable.

1.1 The Information Seeking Paradigm

An information seeking process begins with a user's goals. Let us assume a user who has an information *request* or *interest*. The purpose of an automated information seeking system is to process information *sources* and provide the user with the information sought. An information seeking system consists, in general, of four basic components:

- a way for representing the information request,
- a way for representing information sources,
- a way of comparing the above representations, and
- ways of using the results of the comparison.

Information sources are entities which contain information in a form that can be interpreted by the user. These may contain text, audio, still or animated images. Information sources are commonly referred to as *documents*. Requests and documents are represented by some characterization language. The process of creating representations is widely known as *indexing*, and its goal is to assign *terms* that are deemed to best represent content. In general, indexing can be broken down into *term selection* and *term weighting*. Representation makes it possible to automate the process of computing comparisons, e.g. *scores*, between requests and documents. The results of the comparison are used to display the information sought by the user, or modify the representation of the request — through a *relevance feedback* mechanism — attempting to improve the search. Retrieved documents may also make the user change her mind, so that she issues another (revised) request. A general model of an information seeking process is depicted in Figure 1.1 (Belkin and Croft, 1992; Croft, 1993).

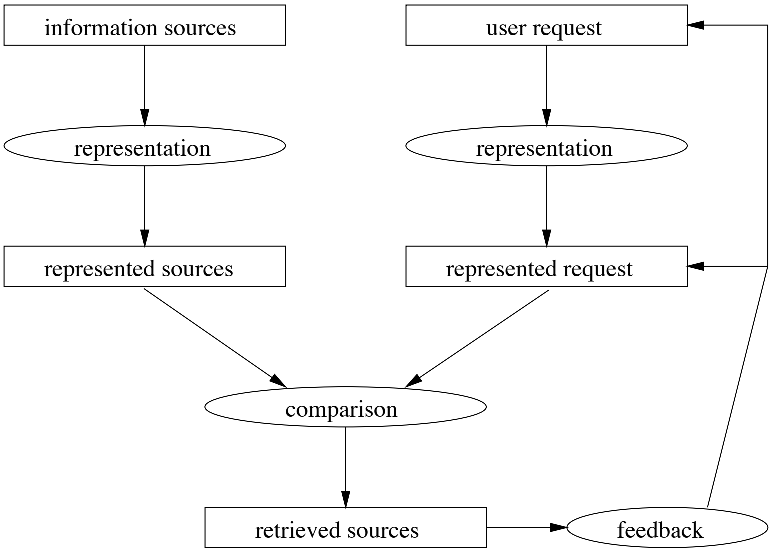


Figure 1.1: A general information seeking model.

The model we have just described is very general. The more precise nature of an information seeking process is determined by the different possible types of information requests and sources. This thesis is mainly about a particular type of information seeking process known as *document filtering*.

1.2 Document Filtering

Document filtering is an information seeking process that searches through a dynamically generated document collection, e.g. a *stream* of arriving documents, for documents which match a user interest. The user interest is assumed to be long-term, in contrast to one-time queries in traditional information retrieval, and we will call it a *topic*. Filtering may also be seen as a *binary classification/categorization* task where each new document has to be classified under one of two categories: relevant, or non-relevant.

Document filtering, and similarly other information seeking processes, can be broken down into three sequentially-performed sub-tasks or modules: *collection*, *selection*, and *display* of documents. The overall picture is shown in Figure 1.2. Collection is concerned with providing a document stream. Two ways of collecting documents may be distinguished: *passive* collection e.g. from a newswire (Denning, 1982), or *pro-active* collection e.g. with autonomous intelligent agents going out to find new documents in the World-Wide Web (Simons et al., 2000). The combination of both actively and passively collecting documents in one stream is also possible. The display module is responsible

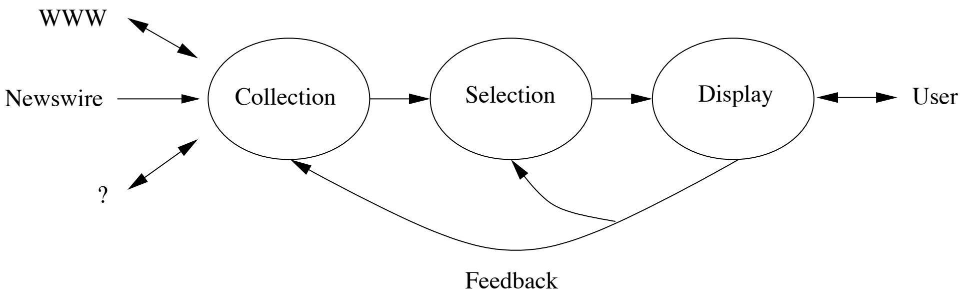


Figure 1.2: Sub-tasks of a filtering process.

for the interaction between users and the system. It not only displays the selected documents, but it interacts with users and accounts for their reactions on the presented output to guide the collection and selection processes. In this study, we focus our attention on the selection module. The collection and display tasks are already rich research areas in their own right, and will be considered here as black-boxes, where the former provides a document stream and the latter provides relevance judgments for some of the selected documents.

The selection module does the actual filtering of the collected documents, selecting the relevant ones or rejecting the non-relevant ones, with respect to a topic. It uses some internal *representation* for documents and topics, called *profiles*. Representation allows, by means of a *filtering function*, the calculation of the aboutness of each document with respect to a topic, so as to decide whether to select or reject it. Two sources of deducing representations have been dominating the research in filtering, distinguishing two types of filtering: *collaborative* (or *social*), and *content-based* (or *cognitive*) (Denning, 1982; Malone et al., 1987).

In collaborative filtering, documents are represented by annotations made by their prior readers. By exchanging these annotations, groups of users with shared interests can automatically be identified. Collaborative filtering can provide a basis for selection of documents regardless of whether or not their content is represented. Content-based filtering on the other hand assumes that each user operates *independently*. There is no exchange of information of any kind, thus document representations can only be derived from their content. Of course, both approaches may be combined in such a way that annotations and content both contribute to estimate the aboutness. In this study, we are concerned with content-based filtering.

Filtering systems can exploit the long-term nature of topics to improve the filtering model over time. A system may continuously monitor the stream, accumulating different kinds of statistical data, and using them to produce better representations for profiles. Moreover, as documents are filtered for a topic, the user may give relevance judgments for some of the selected documents. Judged documents can be used to adapt the topic profile and the filtering function. The choice between exploiting the long-term nature of topics or not, distinguishes between two types of systems. Systems that do not change the way they filter over time are called *batch*[1] or *non-adaptive*. Single-pass filtering systems that alter their filtering model in response to the history are called *adaptive*.

[1] Adhering to the TREC jargon (Chapter 5).

1.3 Filtering vs Retrieval

Information retrieval (IR), and especially text retrieval, is an information seeking process with an extensive research heritage. Given the shared similarities between many information seeking processes, the filtering task has been viewed as a special retrieval case and treated by retrieval techniques. In some cases, the filtering and retrieval tasks have even been seen as "two sides of the same coin" (Belkin and Croft, 1992). As a result, qualitative differences of filtering are usually overlooked. In Table 1.1, we point out a few differences in the nature of data involved.

	retrieval	**filtering**
information request	short-term	long-term
information sources	static	dynamic
comparison	ranking of documents	binary decision
feedback	usually one-time	repeatedly

Table 1.1: Main differences between retrieval and filtering.

Filtering systems can exploit evidence about the relatively long-term interests of a user to develop more complete and precise descriptions of the request. The repeated feedback of information about relevance may result in better representations of the information request, achieving better filtering effectiveness.

The dynamic nature of sources can be seen as documents associated with their time of arrival, i.e. a *stream* of documents. At any point in time, there are documents that have not been seen, thus judgments can only be made based on documents already seen by that time. Furthermore, since documents arrive one at a time, document representations should be built *on-the-fly*.

The dynamic nature of sources differentiates, moreover, the way of comparing them to requests. Where retrieval systems return a *ranked* list of documents (most relevant first, least relevant last), filtering systems should make a *binary decision* whether to accept or reject an incoming document as it arrives. Moreover, while retrieval is about selecting relevant documents, filtering is not only about selecting but it can also be about eliminating non-relevant documents from a stream. Despite these differences filtering has still a lot to borrow from retrieval.

1.4 Themes and Structure of this Thesis

This thesis is a collection of the work that the author has accomplished in the context of the Profile project (Simons et al., 2000). Profile was a multi-disciplinary project aiming at the development of a pro-active filtering system, as an effective intermediary between users and information sources. It has been essentially a research project, giving the opportunities to elaborate on different problems within the general context of information filtering.

Two main themes have been considered by the author, splitting this thesis in two distinct parts which may be read independently:

Document Filtering. The traditional view of filtering as a special case of the retrieval task, we believe, is not appropriate.

In Chapter 2, we view filtering as an *adaptive* and *temporally-dependent* process. A process that, in contrast to traditional retrieval, takes the dynamic nature of relevance and its temporal aspects into account. The chapter is a theoretical study that has resulted from the bottom-up approach we have followed to deal with filtering during the last few years. Guided by the experiments we have performed — in the context of the TREC-9 filtering tasks and elsewhere — we formulate what we believe is important for effectiveness as well as for efficiency. The chapter results in a coherent view on filtering, but the ideas also apply to other information seeking tasks that may involve temporally-dependent data. In Chapter 3, we investigate the use of *time distributions*. Our main hypothesis has been that data which occur uniformly in time give better predictions of the future, thus are more valuable. We test this idea in an indexing context by introducing a novel term selection method, namely the *term occurrence uniformity (TOU)*. In Chapter 4, we provide a novel thresholding method, namely the *score-distributional (S-D) threshold optimization*. Thresholding is required in order to force a binary decision on retrieving a document or not. The problem is important since it has proved critical for the effectiveness of classification tasks (Lewis, 1995a; Hull and Robertson, 1999; Robertson and Walker, 2000).

The total effort put into these directions has resulted in a number of publications (Arampatzis et al., 2000c; Arampatzis and van Hameren, 2001; Arampatzis and van der Weide, 2001), and in the development of the prototype FilterIt system (Arampatzis et al., 2000a). The system has readily demonstrated, in the context of the TREC-9 filtering track (Chapter 5), the feasibility of the ideas and the benefits that they provide to effectiveness. For most of the techniques described in the aforementioned chapters, we pay special attention to practical implementational aspects, such as that of *incrementality*.

Representation of Textual Information. What is a suitable and effective representation for information seeking tasks has been a rather long-standing issue. In its simplest form, representation takes the form of *bag-of-words*, while the more complicated and ambitious ones have not yet established a clear benefit.

In Chapter 6, we discuss *linguistically motivated indexing (LMI)* approaches. We give an overview of the most important attempts to break out of the bag-of-words representations. Guided by the failures and successes of previous approaches, we provide an LMI scheme which deals with language in a coherent and compact way. Chapter 7 describes our experimental work in using linguistic resources and processors for information seeking tasks. It sets out to evaluate parts of our proposed LMI scheme.

This line of research has also resulted in a number of publications (Arampatzis et al., 1997a; Arampatzis et al., 1997b; Arampatzis et al., 1998; Arampatzis et al., 2000b; Arampatzis et al., 2000d).

An implicit contribution of our work is that we evaluate our ideas mainly in *relevance feedback* environments, i.e. environments which involve training data.

The approach to both themes has been theoretical as well as experimental. For readability reasons, usually we provide within the main part of this book only the final formulae and representative samples of empirical evaluations; long proofs, raw experimental results, and additional plots can be found in the appendices. This thesis does not provide an extensive review of all literature in the field; we review only the most related research on the way, and as far as it is necessary for assimilating our work.

The choice of the underlying retrieval model has been quite traditional. We have used the *vector space model*, for its ease of understanding and implementation, and we have built upon it. The same holds for term weighting schemes; we have merely applied existing and proven ones. We believe, however, that these choices do not invalidate our evaluations, and our results are likely to be reproduced using other settings. Our main test-bed — retrieval model, effectiveness measures, and test data — is described in Appendix A; the reader familiar with traditional choices need not read it. Special modifications, settings, and novel techniques are reported where they apply.

Chapter 2

Filtering as an Adaptive and Temporally-dependent Process

The filtering task has traditionally been defined as a special case of the information retrieval task, and undeniably, it can be performed by applying retrieval techniques. This theoretical study summarizes our experiences in viewing filtering as an adaptive and temporally-dependent process. A process that, in contrast to traditional retrieval, takes into account the dynamic nature of relevance and its temporal aspects. We investigate the nature of user interests, formulate useful types of adaptivity, and discuss the effectiveness of those types in relation to user interests. To deal with drifts, we introduce the notion of the *half life* of documents. Furthermore, we discuss potential dangers for effectiveness such as selectivity traps. We pay special attention to practical efficiency issues by discussing term selection and incrementality. This chapter is based on our work previously published in (Arampatzis and van der Weide, 2001).

2.1 Introduction

Information retrieval, and especially text retrieval, is an information seeking process with an extensive research heritage. Given the similarities shared between many information seeking processes, the filtering task has been seen as a special retrieval case, treated by retrieval techniques. In some cases, the filtering and retrieval tasks have even been seen as "two sides of the same coin" (Belkin and Croft, 1992). We do not question the similarity of the tasks; the filtering task can indeed be performed with slightly modified retrieval techniques. However, we point out a few important differences in the nature of data involved. Taking these differences into account is beneficial for effectiveness.

This study is influenced by the work of several researchers. We have found especially useful the conceptual framework for filtering described in (Oard and Marchionini, 1996), and the adaptivity issues discussed in (Williams, 1991). We additionally refresh and revise the most important parts of our work described in (Arampatzis et al., 2000c; Arampatzis et al., 2000a).

In the following sections, document filtering systems are addressed by:

- classifying user interests with respect to how the idea of relevance changes over time (Section 2.2). As we will see, relevance may be disturbed by user-triggered and world-triggered factors.

- classifying user interests with respect to the occurrence patterns of relevant documents in time (Section 2.3). We introduce a measure which enables the temporal classification of interests. Moreover, we outline how such information may be used in filtering.

- classifying forms of adaptivity (Section 2.4).

- discussing implementation issues, such as incrementality (Section 2.5).

- discussing the performance of different forms of adaptivity on different kinds of user interests (Section 2.6).

- discussing term selection for adaptive filtering tasks (Section 2.7).

- discussing potential dangers for effectiveness, such as selectivity traps (Section 2.8).

This study is the result of the bottom-up approach we have followed to deal with filtering in the last years. Guided by the experiments we have performed — in the context of the TREC-9 adaptive filtering tasks and elsewhere — we will formulate what we believe is important for effectiveness, as well as, for efficiency.

2.2 A Relevance Classification of Topics

A filtering task begins with a user interest and a stream of documents. With respect to a stream of N documents, and assuming binary relevance, we will define as *topic* T the substream of all documents relevant to the user's interest, e.g., $T = D_1, \ldots, D_n,\ n \leq N$. This definition of topic quantifies the user interest in terms of the document stream. We will assume that the topic is *persisting* in the stream, that is, as the stream grows ($N \rightarrow \infty$) the topic grows as well ($n \rightarrow \infty$).

Adopting this point of view, only 2^N different topics may be distinguished for a certain N, however, an infinite number of interests may be thought of. When two or more different interests translate to the same substream of relevant documents, we will not distinguish between those interests; the idea is that you cannot get anything more than what is actually present in the stream.

Let us assume an *abstract distance measure* $d(D_i, D_j) \in [0, +\infty)$ between any two documents D_i, D_j. Small distance values mean that two documents are about similar *subjects*. We will also introduce a *fuzziness parameter* ε which denotes the maximum distance allowed for two documents to be considered as being about the same subject.

A topic T may be classified with respect to the values of the distance $d(D_i, D_j)$, for all relevant documents D_i, D_j, as $n \rightarrow \infty$:

- **stable:** All distances between the documents are less than or equal to ε:

$$d(D_i, D_j) \leq \varepsilon \ , \quad \forall i, j \ .$$

- **drifting:** All distances between consecutive documents are less than or equal to ε, but some distances of non-consecutive documents are not:

$$d(D_i, D_{i+1}) \leq \varepsilon \ , \quad \forall i$$

and

$$\exists i, j : \ d(D_i, D_j) > \varepsilon \ .$$

- **multimodal:** There are consecutive document distances greater than ε, but the topic can be broken down to a finite number of k stable disjoint subtopics:

$$\exists \ \text{stable} \ T_1, \ldots, T_k : \ T = \oplus_i T_i$$

and

$$d(D_i, D_j) > \varepsilon \ , \quad \forall D_i \in T_l \ , \ \ \forall D_j \in T_m \ , \ \ \forall l \neq m \ ,$$

where $\oplus$ means that T_i's are *exclusive partitions* of T: they have no documents in common but their union amounts to T.

- **vagrant:** the same as multimodal, but the number k of subtopics is infinite.

- **white noise:** the same as vagrant, but $k \rightarrow \infty$ faster than for vagrant topics.

This classification is rough, but sufficient for our analysis. A topic may exhibit in reality a more complex behaviour in time by switching between two or more of the above types. For example, a topic is at first stable, but then starts drifting; or even a subtopic T_i of a vagrant topic is drifting.

Note that the fuzziness parameter ε determines the limits of the classes: a very large fuzziness will classify all topics as stable, while an infinitesimal one will classify everything as white noise. However, for a given reasonable fuzziness, what classifies a topic under one of the above categories depends on *user-triggered* and *world-triggered* factors.

User-triggered factors are related to whether a user interest shifts in time, and how it shifts. World-triggered factors are independent of shifts in user's interest. They are directly related to the nature of the interest with respect to the real world. The world may produce considerably different but still relevant documents.

2.2.1 User-triggered Shifts in Interest

A user who sticks to her initial request has a *stable interest*. However, the user interest can also deviate over time. For instance, as the user reads more and more documents about the initial request, she wants to know more specific or general information, or slowly becomes interested in a similar subject which is referred to in the documents already retrieved. In this case, the user has an *drifting interest*. (Allan, 1996) has demonstrated that such drifts can be handled readily by phasing out old context.

A *multimodal* or *vagrant interest* usually arises when the user does not exactly know what she is looking for, consequently the interest is vaguely formulated. She will probably find different kinds of documents relevant, in the search for her real interest. The interest may switch between closely related — specific or relatively random — domains.

We will assume here a rational user who does not abruptly change her mind. An abrupt shift should be considered as a different interest and be treated as a new filtering process. Thus, white noise behaviour can not arise for user interests in filtering; it rather corresponds to user interests in traditional retrieval tasks.

2.2.2 World-triggered Shifts in Document Content

Consider filtering an interest about *HIV treatments*. Over the years, treatments have changed; new and more effective ones have been slowly developed, while the less effective ones have been fading out. In such cases, where the contents of relevant documents slowly change in time, there is *content drift*.

Document contents can show multimodal or vagrant characteristics. Multimodality arises when the interest is such that it combines two or more stable but relatively distant interests, for example, *operating systems* AND *computer architecture*. The contents of relevant documents will switch between the two different subjects at irregular intervals.

A special kind of vagrancy arises in what we call *event-driven interests*. As an example consider the interest *terrorism*. Such an interest is driven by real-world *events* which can be relatively different and unexpected, for example, *NYC subway bombing* or *flight TR-304 hijacking*. An important event is usually associated with bursts of relevant documents for some period of time. Then, documents about the subject tend to disappear completely from a news stream, while some other (relatively random) terrorist event may happen.

2.2.3 Relevance

User interest shifts and *document content shifts* are related in the sense that the idea of relevance changes. Whether a shift comes from the user or the world side is not of importance. What is important is that future relevant documents will be different than the ones of the past. Consequently, we will talk about *relevance shifts*, irrespective of who or what causes them.

The user and the world can both be viewed as sources of disturbances for relevance. In this respect, the source with the highest *entropy* defines the class of the topic. For example, a stable user interest but vagrant contents in relevant documents results in a vagrant topic.

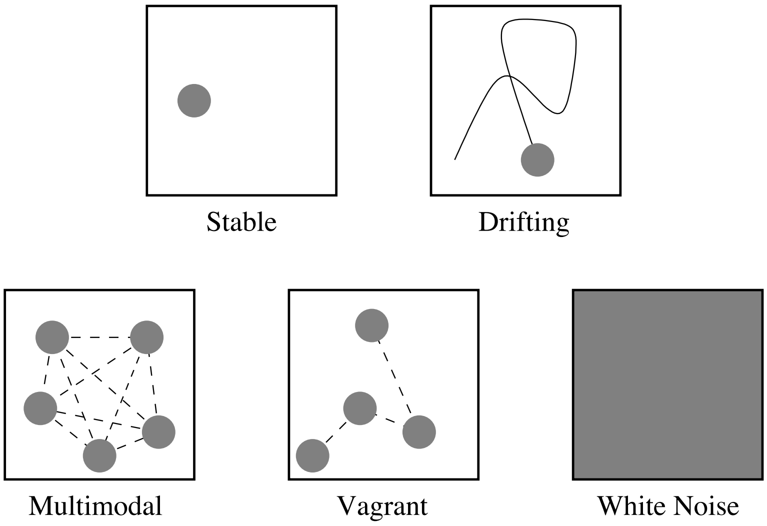


Figure 2.1: A relevance classification of topics.

In summary, an interest is what a user has in mind. An interest may be satisfied by a number of (finite or infinite) subjects. A document contains a few subjects. The same subject may spread across documents. A topic is the substream of all documents containing subjects that satisfy the interest at the time of their arrival. With respect to how relevance changes, i.e., how the contents of relevant documents change in time, topics may be classified as shown in Figure 2.1. The figure shows possible trajectories of relevance in the document space.

The classification of topics we have just considered is related to the types of adaptivity we will introduce in Section 2.4. In Section 2.6 we will discuss this relationship. First we will attempt another classification of topics.

2.3 A Temporal Classification of Topics

The classification of topics considered in the previous section is purely based on relevance aspects. We have considered how relevance changes in the ordered set (stream) of a topic's relevant documents. In this section, we consider the actual times of arrival of relevant documents. The qualitative classification we suggest consists of the following classes:

- **simply periodic:** Single relevant documents arrive at approximately constant time intervals.
- **random** or **uniform:** Relevant documents arrive at irregular intervals.
- **periodically clustered:** Some relevant documents arrive at regular time intervals.
- **aperiodically clustered:** Bursts of relevant documents arrive at irregular time intervals.

Figure 2.2 depicts the above occurrence patterns. This classification of topics is rather orthogonal to the relevance classification considered in Section 2.2.

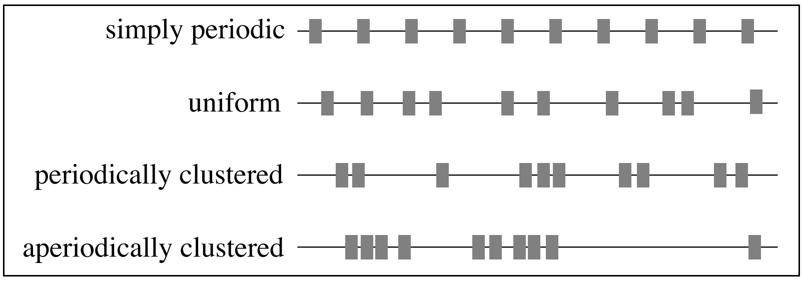


Figure 2.2: A temporal classification of topics.

Next we will see how *uniformity* may be quantified. The measure we will introduce enables the temporal classification of topics as discussed above. Then we will briefly discuss the implications that such a classification has for filtering effectiveness.

2.3.1 A Measure for Uniformity

Let us consider a *normalized time-line* $[0, 1]$, where the initiation of a filtering task is located at 0 and the present time is at 1. Each document occurrence can now be represented by a point in that interval, and the occurrence pattern of a topic of length n by a list of points $x_1, \dots, x_n$. Measures of non-uniformity of point-lists are called *discrepancies*. Such measures have the structure of statistics to measure the overall difference between an estimated probability distribution and a conjectured probability distribution.

A list of n occurrence points can be converted to an unbiased estimator $S_n(x)$ of the cumulative distribution function of the probability distribution from which it was drawn: $S_n(x)$ is the function giving the fraction of occurrences to the left of x. The cumulative distribution function of the *uniform distribution* is $P_U(x) = x$. Different lists of points have different cumulative distribution function estimates. However, all cumulative distributions agree for $x = 0$ and $x = 1$ where they are zero and one respectively. As a consequence, it is the behaviour between 0 and 1 of their cumulative distribution functions that distinguishes distributions.

There are many statistics to measure the overall difference between two cumulative distributions. We have chosen a variant of the generally accepted *Kolmogorov-Smirnov (K-S) test*, namely *Kuipers' statistic* (Kuipers and Niederreiter, 1974), which is the sum of the maximum distances of $S_n(x)$ above and below $P_U(x)$:

$$V_n = D_+ + D_- = \max_{0<x<1} [S_n(x) - P_U(x)] + \max_{0<x<1} [P_U(x) - S_n(x)] \,. \tag{2.1}$$

The method is demonstrated in Figure 2.3.

This statistic guarantees equal sensitivities at all values of x, in contrast to the original K-S test which tends to be more sensitive around the median value where $P_U(x) = 0.5$ and less sensitive where $P_U(x)$ is near 0 or 1. It is also invariant under re-parameterizations of x and shifts on the circle created by gluing together points zero and one of the time-line. K-S-like statistics have a computational complexity linear to n. More details on how to compute them can be found in (Press et al., 1992).

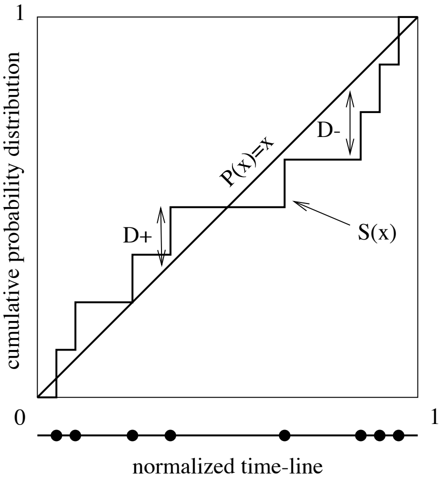


Figure 2.3: Kuipers' discrepancy test $V_n = D_+ + D_-$.

V_n takes values in $[\frac{1}{n}, 1]$. Figure 2.4 shows the empirical probability densities of V_{10} and V_{100} for 20.000 pseudo-random occurrence patterns. Values close to $1/n$ are obtained for simply periodic occurrences. Truly random patterns get slightly larger values; how much larger depends on the number of occurrences n, as it is shown in Figure 2.4. Values of V_n close to 1 correspond to serious clustering of the occurrences in the timeline.

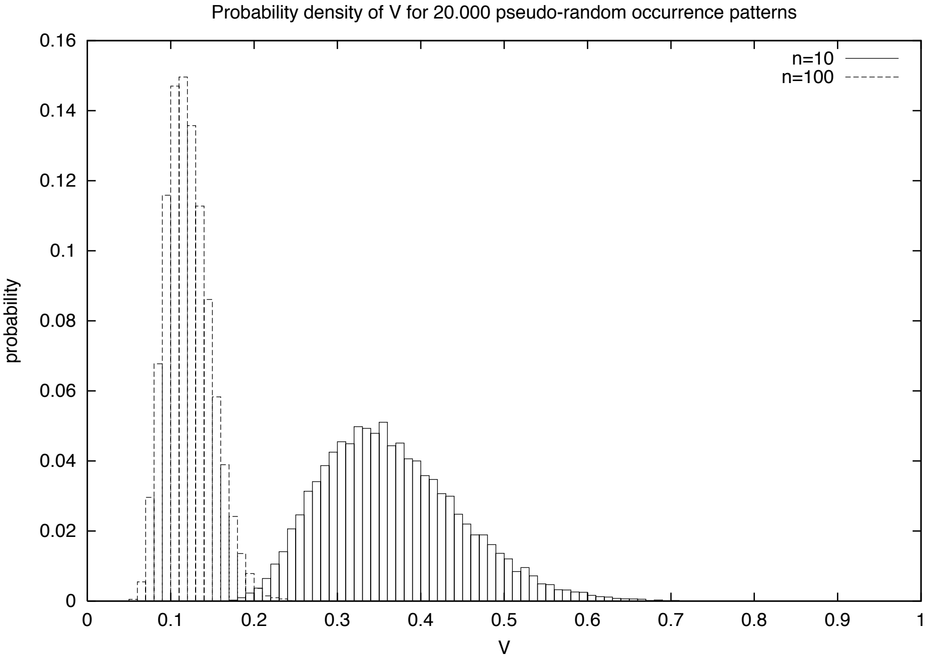


Figure 2.4: Probability density of Kuipers' statistic for uniform occurrence patterns.

Using Kuipers' statistic, topics may be quantitatively classified into the classes defined at the beginning of this section. We simply split the range of values $[\frac{1}{n}, 1]$ of V_n into four intervals. These intervals are determined by three values x_1, x_2, and x_3. For a certain n, we define x_1 and x_2 through the equations

$$P(V_n < x_1) = p \quad \text{and} \quad P(V_n > x_2) = p \; , \tag{2.2}$$

for some small p, e.g. $p = 1\%$. For a certain p, x_1 and x_2 may be obtained from standard tables with the confidence levels of the statistic, e.g. from (Knuth, 1981). Moreover, we define x_3 as some number between x_2 and 1. its exact value is a rather subjective matter and it should be justified empirically.

2.3.2 An Example of Topic Uniformity

Table 2.1 gives the uniformity of occurrences of relevant documents in time for categories of the Reuters-21578 corpus (Section A.6.1). Only the categories with at least 100 relevant documents in the training part of the stream are presented.

topic	n = training docs	test docs	$1 - V_n$
earn	2861	1087	0.813
acq	1648	719	0.827
money-fx	534	179	0.769
grain	428	149	0.881
crude	385	189	0.794
trade	367	117	0.739
interest	345	131	0.794
wheat	211	71	0.866
ship	191	89	0.859
corn	180	56	0.831
money-supply	132	34	0.815
dlr	131	44	**0.581**
sugar	125	36	0.815
oilseed	124	47	0.824
coffee	111	28	0.806
gnp	101	35	0.771

Table 2.1: Uniformity of Reuters-21578 topics.

The training stream covers a period of 40.4 days, calculated from the time of arrival of the first training document of any topic to the first test document. The rightmost column gives the *topic occurrence uniformity*, or just *topic uniformity*), in the training stream. The closer this number is to one, the more constant the delivery rate (relevant documents per time unit) for that topic. Obviously, documents about *dlr* (dollar) arrive in bursts, a possible consequence of temporal events concerning e.g. dollar's exchange rate drops in Tokyo.

Figure 2.5 gives the temporal histograms of document arrivals for topics *dlr* and *money-supply*. We can see that many of *dlr* documents arrive in the period of days 25 and 37, while the document distribution in the time-line for topic *money-supply* is more uniform. A comparison of their cumulative distribution functions to the cumulative distribution function of the uniform distribution is given in Figure 2.6.

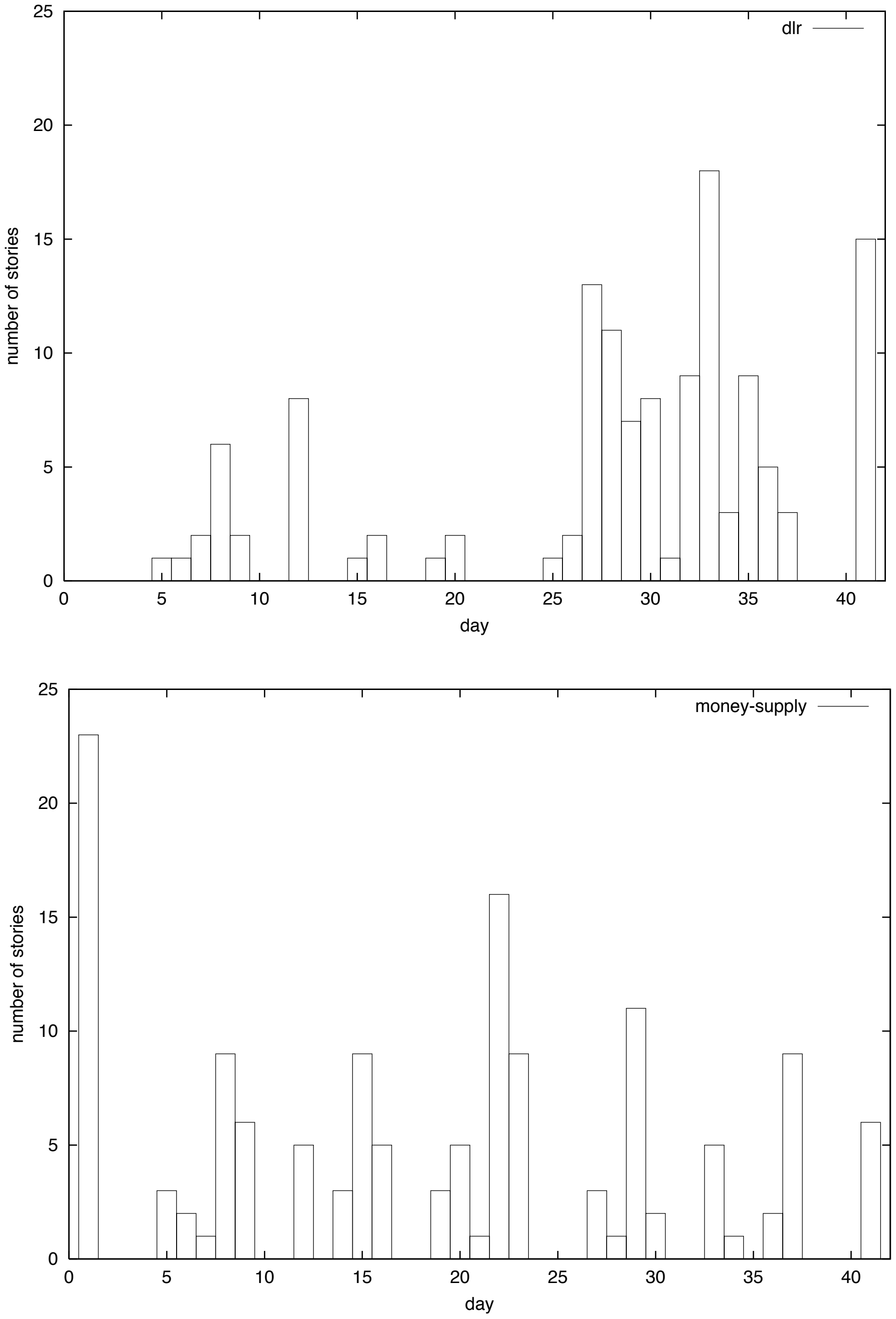


Figure 2.5: Temporal histograms of document arrivals for 2 topics.

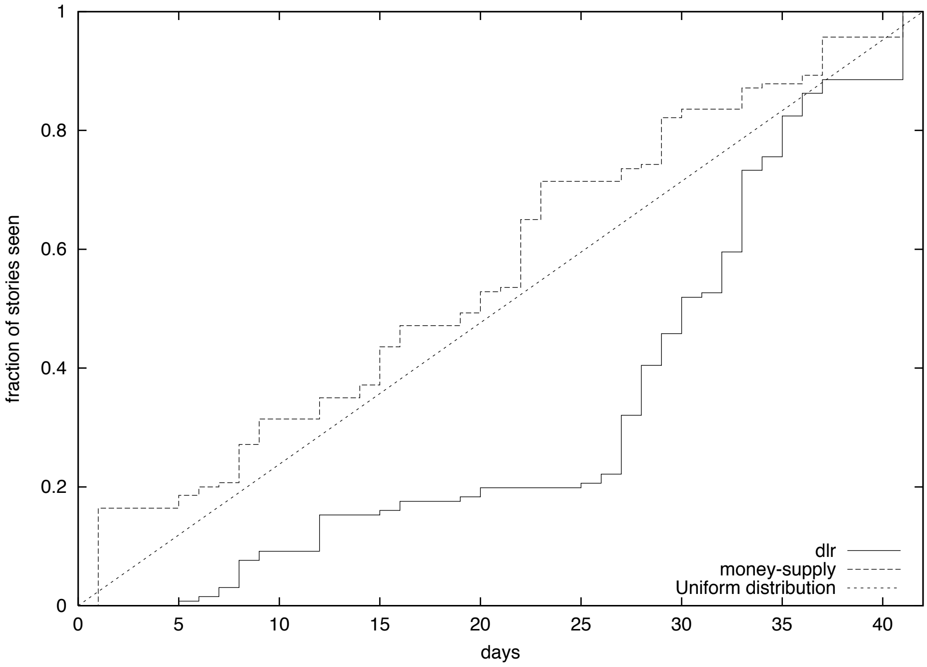


Figure 2.6: Cumulative distribution functions of document arrivals for 2 topics.

The 2-day gaps (e.g. days 10-11, 17-18, etc.) in *money-supply* correspond to weekends where no economic news is made. Note that this match is not exact since the days in the plots do not correspond to real days; they are successive 24-hour intervals taken from the arrival time of the first document of the training stream. Considering also the different closing times of international stock markets, it should explain why few economic documents seem to occur in weekends, especially for *dlr*. We will report experiments with time distributions on Reuters in Chapter 3.

2.3.3 Using Temporal Information

We will outline how information about the occurrence pattern of a topic in time may be used for filtering. Let us consider again a stream of N documents and a topic T of length n. The *density* of relevant documents in the stream for T is

$$\rho = \frac{n}{N} \,. \tag{2.3}$$

If the topic occurs randomly in the stream, then ρ may be interpreted as the probability that the next arriving document will be relevant. However, high topic uniformity is not the case in general. Periodic and clustering characteristics introduce uncertainty into the interpretation of density as probability. The uncertainty decreases with the topic uniformity.

V_n, ρ, and periodicity information may provide means for filtering irrespective of document content. ρ can be seen as the *expected value* of the *a priori* probability of relevance $P(\text{rel})$, i.e., $\mathsf{E}(P(\text{rel})) = \rho$. The variance of the distribution of $P(\text{rel})$ in time is some increasing function f_n of V_n, i.e., $\mathsf{V}(P(\text{rel})) = f_n(V_n)$. Periodicity information may give an estimate of $P(\text{rel})$ that corresponds to a certain time-point t:

$$P(\text{rel}|t) = \mathsf{E}(P(\text{rel})) + g(\mathsf{V}(P(\text{rel})), t) \; , \tag{2.4}$$

where g is some function that accounts for periodicity and/or temporal clustering.

In principle, one could blindly retrieve documents by sampling the document stream with probability $P(\text{rel}|t)$. $P(\text{rel}|t)$ is usually small, since it depends on ρ which is small because there are usually many more non-relevant than relevant documents in a stream. However, depending on the temporal nature of the topic, $P(\text{rel}|t)$ may peak at usable values. In any case, $P(\text{rel}|t)$ may be seen as additional evidence that together with $P(\text{rel}|D)$ (the probability of relevance estimate based on document content) contributes to the decision of whether to select a document or not.

What we have just described is rather crude, and we do not claim that this is the best way to deal with the temporal aspects of filtering. Summarizing the problem, the questions are: How can $P(\text{rel}|t)$ be estimated for the history? How can one extrapolate $P(\text{rel}|t)$ for the future? What is the appropriate way to combine the two pieces of evidence $P(\text{rel}|t)$ and $P(\text{rel}|D)$? In fact, the tools are already there; here are a few keywords: *Fourier analysis*, *time-series analysis*, or more contemporary and geometrically, *phase space reconstruction* and *Poincaré sections*.

2.4 A Temporal Classification of Adaptivity

Disregarding the actual techniques used for creating or altering a filtering model, filtering systems may be classified according to the *temporal location* from which they obtain the information for doing that. To reach such a classification we will follow an approach similar to the one in (Williams, 1991).

Let us consider a system that is initiated at time 0 and the current time is t; thus the system has a history of length t. The importance of an *event* that happened at time x within this history can be modeled by a *history weight function* $H(x,t)$ with the following property:

$$\int_0^t H(x,t)\, dx = 1 \; , \quad \forall t > 0 \; , \tag{2.5}$$

that is, the area below the $H(x,t)$ curve amounts always to 1 for all t. For instance, a history weight function that weighs equally all history is:

$$H(x,t) = 1/t \; . \tag{2.6}$$

Irrespective of its form, the $H(x,t)$ curve is characterized by its mean value, which is mathematically defined as:

$$\bar{H}(t) = \int_0^t H(x,t) x \, dx \; . \tag{2.7}$$

It will be useful for this analysis to define the distances $a(t)$ and $b(t)$ of this mean from the beginning and the end of the history respectively:

$$a(t) = \bar{H}(t) \; , \quad b(t) = t - \bar{H}(t) \; . \tag{2.8}$$

Figure 2.7 visualizes all the above so far.

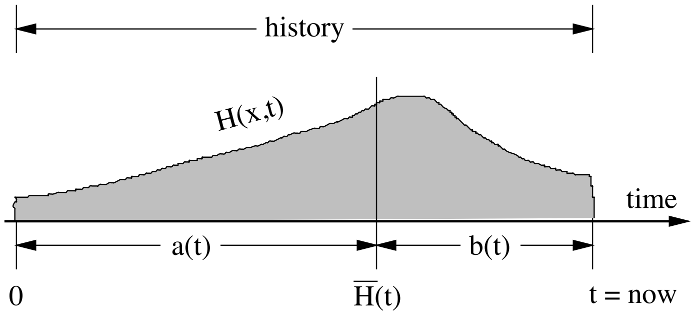


Figure 2.7: An example of a history weight function.

Adaptive systems may be classified according to the behaviour of $\bar{H}(t)$ as the history grows, that is $t \to \infty$. We distinguish between the following classes of adaptivity:

- **non-adaptive:**

$$a(t) = 0 \ , \quad b(t) \to \infty \ .$$

- **locally adaptive:**

$$a(t) \to \infty \ , \quad b(t) < c \ ,$$

  where $b(t) < c$ means that $b(t)$ is bounded by a constant c as $t \to \infty$.

- **asymptotically adaptive:**

$$a(t) \to \infty \ , \quad b(t) \to \infty \ .$$

Non-adaptive systems do not use the history whatsoever. Asymptotically adaptive systems spread the emphasis over the whole time-line in such a way that the mean $\bar{H}$ is not bounded. An example of an asymptotically adaptive system is a system which weighs all events of the past equally, like one with a history weight function of Equation 2.6.

Locally adaptive systems rely most heavily on data collected in the recent past, degrading the value of the early past as the history grows. A minimum amount of emphasis is always given to a bounded length of the recent history, and the rest of the emphasis is spread over the rest of the history. A special case of local adaptivity shows up in *windowed locally adaptive* systems which consider only the recent history within a fixed time window. A typical history weight function of this form is:

$$H(x,t) = \begin{cases} 1/W \ , & \text{if } \ t - W \leq x \leq t \ . \\ 0 \ , & \text{if } \ 0 \leq x < t - W \ . \end{cases} \tag{2.9}$$

where W is the window size.

In Section 2.6, we will discuss the effectiveness of the aforementioned types of adaptivity in relation to the nature of user interests. First, turning the theory into practice, we will discuss some practical issues in implementing adaptivity.

2.5 Incrementality

Our discussion so far has assumed that the whole history and an unlimited amount of memory and computational power are available at any point in time. However, practical models in order to be feasible should satisfy the following requirements:

- use a fixed finite amount of memory.

- process the available history in a fixed finite number of computations.

These requirements imply that only a finite portion of the history should be retained, and that models should be implemented *incrementally*.

For the sake of the discussion, let us assume a simple filtering model that records frequencies of certain features occurring in relevant documents, in order to make predictions of relevance in the future. Incremental asymptotic adaptivity in such a simple model can be achieved by accumulating the values of the occurring features in an array of registers; one register per feature. Of course, there is another minor concession we make here, that is to allow registers of infinite width. Double precision arithmetic approximates this assumption sufficiently; in any case, all accumulators can be divided by a constant, whenever a value approaches the maximum width of the registers, without invalidating the model.

A locally adaptive system may be implemented in a similar manner by additionally maintaining a document buffer of some length W. Every time a new document arrives, registers accumulate the values of the occurring features, but they are also decremented by the values of features which occur in the oldest document in the buffer. This approach is incremental, but it has two disadvantages: it uses more memory because of the document buffer, and it discards all information beyond what is in the buffer at any time.

An alternative approach, which uses all information but weighs it appropriately, is to perform a *decay* operation. We define the *half life* h of a document as the age that a document must be before it is half as influential as a fresh one. If a document D_i has arrived at time t_i and the current time is t_n, the history weight of the document is:

$$l_i = \exp\left(\frac{\ln 0.5}{h}(t_n - t_i)\right), \tag{2.10}$$

where t_n, t_i, and h are expressed in the same units, e.g., months. Figure 2.8 demonstrates the decay operation.

The decay operation can be performed incrementally, and it does not require any document buffers. It is easy to show that when D_n arrives, all accumulators have only to be multiplied by l_{n-1} before the new values of the features occurring in D_n are added.

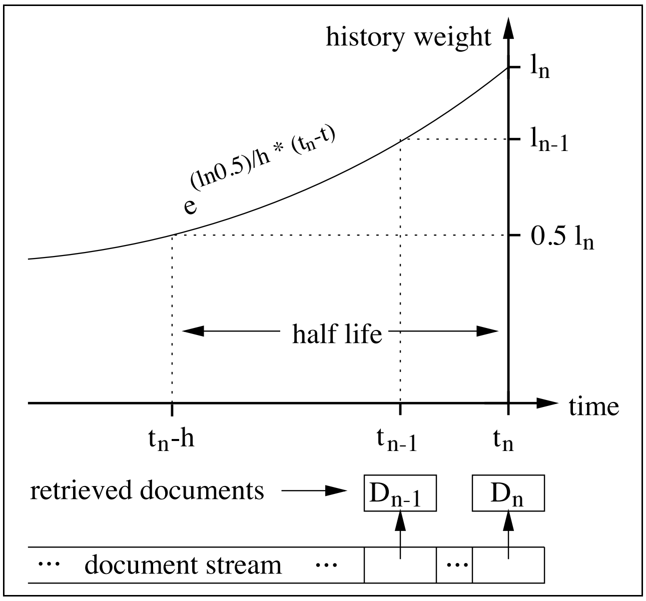


Figure 2.8: Decay and half life.

2.6 A Comparison of Adaptivity

Non-adaptive systems will perform reasonably as long as the initial topic representation is complete and precise and the topic is stable. However, the initial representation is bound to be incomplete and imprecise, due to two factors:

- the incapability of users to verbalize their precise interest,
- the weaknesses of the representation scheme itself.

Consequently, locally and asymptotically adaptive systems present more interesting features.

Locally adaptive systems use more recent information and they are capable of responding to relevance shifts quickly. Therefore, they can *track* a drifting topic. However, the disadvantage of them is that they will never converge to a stable topic. Asymptotically adaptive systems have the ability to *converge* to a stable topic. The choice between a locally or asymptotically adaptive system should be made on whether *responsiveness* or *convergence* is more important.

Implicit in the idea of tracking a topic using the history is that the history gives an indication of where the topic may currently be located. A fundamental trade-off exists in tracking topics. While it is advantageous to use as many instances of the history as possible to estimate accurately a topic's position, it is disadvantageous to use outdated instances. Relevant instances of the far past indicate the position of the topic at the time they occurred and they do not reflect the topic's current position. Thus they are less informative than recently occurring instances. A practical solution to this problem is to estimate the speed of a drifting topic and use this estimation to choose an appropriate window size W or half life h.

In the TREC-9 filtering task, the user requests were given as being stable, suggesting that an asymptotic behaviour would be more proper. However, the test stream (OHSUMED) consists of documents collected in a period of five years and it is likely that there are document content drifts. As an example, think of new treatments developed for the same sickness. Indeed, our experiments have shown that the average effectiveness (as this is measured by T9U averaged over all topics) peaks for a half life value of around 4 years (Arampatzis et al., 2000a). Analysis per topic, however, has revealed that effectiveness is optimal at a considerably different half life value per topic. We will report these experiments in Chapter 5.

As a first step in optimizing h per topic, we define the *effective relevance velocity* v of a topic as:

$$v = \frac{d(D_1, D_n)}{n-1} \ , \quad v \in [0, \epsilon] \ . \tag{2.11}$$

Note that the definition considers only the initial and the last position of relevance, and discards the trajectory in between. Moreover, the velocity is defined with respect to the number of steps taken, rather than the actual time. Obviously, h and v are related in an inverse way, however, their more precise relationship should be established experimentally.

The types of adaptivity we have defined are capable of dealing with stable and drifting topics. The question of how multimodal or vagrant topics should be treated still remains. A solution would be to model their subtopics separately. In the multimodal case, all subtopics may be assumed stable and be dealt with by asymptotic adaptivity. However, it may be more effective for the vagrant case to assume that subtopics are drifting. We should remind the reader that the *poles* (the gray circles in Figure 2.1) of a vagrant topic may not be revisited by relevance in the future. Thus, a locally adaptive system would eliminate such old outdated context. Next, we will discuss an alternative way of dealing with multimodality and vagrancy.

2.7 Stabilizing Multimodality or Vagrancy

The solution of modeling subtopics separately is not practical, although it may be effective. A more practical solution is to, first, re-construct the document space so as to bring the different poles as close together as possible, and then assume a larger fuzziness parameter so that the topic may be considered as stable or drifting.

Transformations of the document space may be performed by different techniques, e.g. *feature selection* (Yang and Pedersen, 1997), *latent semantic indexing* (Deerwester et al., 1990), or via the use of *kernel functions* of *support vector machines* (Schölkopf, 1998; Dumais, 1998). Roughly speaking, the idea consists of removing non-informative features (dimensions) and/or constructing new features by combining lower level features into higher-level orthogonal dimensions. We will refer to all methods for transforming a document space as feature selection. Two example transformations are depicted in Figure 2.9.

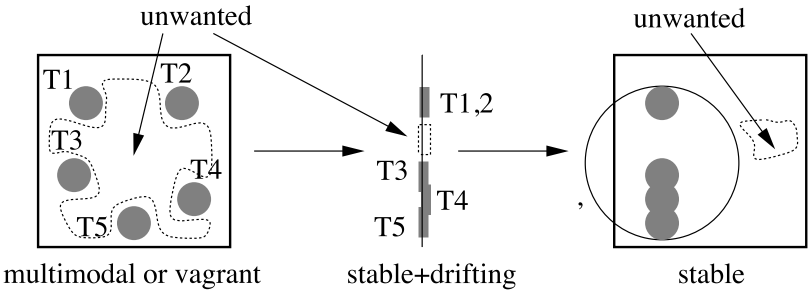


Figure 2.9: Stabilizing multimodality or vagrancy by re-constructing the document space.

The first transformation shows how poles may be brought together by eliminating a dimension. The second transformation shows how unwanted areas of the document space may be moved away from the topic area by adding a dimension (a different dimension than the one eliminated before). By selecting an optimal set of features in this way and by increasing the fuzziness constant (e.g. to ε') if necessary, multimodal or vagrant topics may be treated as stable.

Traditional feature selection schemes usually favour features which occur frequently in relevant documents but infrequently in the rest. In order to eliminate multimodalities or vagrancies, however, it is also important that a feature occurs across poles; these features bring the poles together. High frequency in relevant documents implies that a feature may also occur across poles, but not necessarily.

The uniformity measure we have introduced in Section 2.3.1 may be recruited once more. Based on the hypothesis that features which occur uniformly in time are more valuable than others, we have introduced in (Arampatzis et al., 2000c) a novel feature selection method, namely the *term occurrence uniformity (TOU)*. A small experiment has neither proved nor disproved the hypothesis. The results, however, have been promising, since the method seemed as effective as other powerful term selection methods such as document frequency thresholding[1]. The approach taken has been a brute-force one; candidate features were ranked simply according to their uniformity. A wise integration of a TOU method and some other powerful time-disregarding term selection method may combine the benefits of both approaches. We will report these experiments in Chapter 3.

A fundamental difference between adaptive filtering and classification (non-adaptive) systems is that in filtering the document space may be reconstructed several times in order to optimize effectiveness and efficiency. *On-the-fly* feature selection schemes should be applied with respect to possible relevance shifts. Moderate cutoffs will be more appropriate. Due to the fixed-memory model required for practical systems, every time a cutoff is applied, some low-frequency features will be irretrievably lost. Relevance drifts are associated with frequency increments of previously low-frequency features. Therefore, applying repeatedly aggressive cutoffs will not allow for the tracking of relevance drifts.

[1] Document frequency thresholding has proven to be more than just an *ad hoc* approach for feature selection, and quite powerful in text categorization environments (Yang and Pedersen, 1997).

2.8 Selectivity Traps

The output of traditional retrieval systems is usually a ranked list of documents in order of decreasing scores (given by the probability of relevance or some other similarity measure) with respect to a query. In binary classification tasks, like document filtering, a decision should be made for every document as to whether it belongs to a given class or not. Thus, decisions such as where to “cut” a ranked list have to be made automatically. In some cases, decisions are required to be made as soon as a document arrives, therefore ranked lists are not even possible.

These considerations suggest the *thresholding* of document scores. We will expand on thresholding in Chapter 4. Thresholding has proven to be critical for classification effectiveness and has revealed the twin danger — unique to such environments — of *selectivity traps*: setting a threshold too high retrieves nothing at all, while setting it too low retrieves far too many documents (Robertson and Walker, 2000). We will call these traps *overselectivity* and *underselectivity*, respectively.

Bad thresholding, however, is not the only cause of falling into selectivity traps. Another cause may be training. Usually, a system is trained on its history, i.e. it is trained to do past tasks, and then it is applied to future tasks. Consequently, the success of training depends on whether the lessons learnt from the past apply to the future. The most obvious reason why this might not hold is that a topic is drifting faster than a system is capable of tracking. We will call this trap *intractability*.

Another danger of training is what is widely known as *overfitting* a topic profile on history data. For example, putting too much effort into finding the perfect profile for the history may discover and emphasize accidental characteristics (e.g. typographical errors in relevant documents) that do not generalize into the future. Overfitting usually manifests itself as overselectivity. At the other end of the spectrum lies *underfitting*, which leads to underselectivity. Available training data may not be sufficient for training, subsequently the topic profile is far from convergence describing a bit too much of the document space. Table 2.2 summarizes the possible traps, their causes, and their criticalities for adaptive filtering.

	underselectivity	**overselectivity**	**intractability**
causes	– underfitting – too low threshold	– overfitting – too high threshold	– too fast drift
criticality	not too dangerous	dangerous but recoverable	unrecoverable

Table 2.2: Selectivity traps.

In adaptive filtering, overselectivity is a more dangerous trap than underselectivity. Adaptivity in filtering counts on the system to keep retrieving documents so it can continuously refine the filtering model. In this respect, adaptivity can pull the system out of an underselectivity trap by improving the topic profile and increasing the threshold. On the other hand, if at some point in time the system is led into an overselectivity trap, it will not retrieve any documents on which it can refine the topic profile and threshold, which leads to “silent” profiles. However, such a situation may be recoverable by the use of special mechanisms; the questions are how one can detect and fix an overfitted topic profile, and how can one be sure that the threshold is too high (as opposed to there just being no relevant documents to be retrieved).

In the long term, the intractability trap has essentially the same effect as overselectivity. Even if a profile still retrieves non-relevant documents when it has lost the relevant document area, these non-relevant documents only give an indication of the area that the estimation of relevance should move away from, without specifying an alternative direction. The profile will eventually fall "silent", because of adaptivity responding by increasing the threshold. We prefer, however, to view intractability as a separate trap from overselectivity, since its cause is different and the situation is rather hopelessly unrecoverable.

One should keep in mind that adaptive filtering is an especially sensitive task. What makes it so sensitive is that the system is provided with absolutely no relevance feedback for non-retrieved documents. Any relevance statistics collected in this way are bound to be *partial* in the sense that they do not represent a sample of the whole document space, but a sample of the *retrieved* space, therefore they may be highly misleading. Compare this situation to other adaptive tasks such as adaptive data compression, where the current frequencies of *all* symbols in a channel are known (Williams, 1991).

2.9 Summary

This chapter has summarized our experiences in viewing filtering as an adaptive and temporally-dependent process. All models and ideas we have described are the result of our experimental work in the context of the TREC-9 filtering task (Arampatzis et al., 2000a) (Chapter 5), (Arampatzis et al., 2000c) (Chapter 3), and of previously unpublished empirical investigations, and they result in a coherent view on relevance feedback environments involving temporally dependent data.

We have presented a collection of ideas: a definition of the filtering task, a definition of the topic, two orthogonal classifications of topics (one based on relevance and the other on temporal aspects), a classification of adaptivity, and ways of using temporal information for selecting documents and for feature selection. Moreover, we have discussed potential dangers such as selectivity traps, and paid attention to practical issues such as incrementality. Our analysis has been rough, and we rather pose more questions than provide answers.

The classical view of the filtering task as a special case of the traditional information retrieval task, we believe, is not appropriate. In the last few years, there has been increasing evidence that viewing filtering as an adaptive and temporally-dependent task is beneficial for effectiveness. We are convinced that Information Retrieval in general could benefit by taking the effect of adaptation and time into account. Our work so far is fully described in this thesis and it has concentrated in working out these issues.

Chapter 3

Terms and Time Distributions

This chapter is based on our previously published work in (Arampatzis et al., 2000c). We investigate the use of time distributions in information seeking tasks with relevance data. Specifically, we introduce a novel term selection method, namely the *term occurrence uniformity (TOU)*, based on the hypothesis that terms which occur uniformly in time are more valuable than others. Our current concern is filtering, but this line of research can easily be extended to other retrieval tasks which may involve temporally-dependent data.

3.1 Introduction

In information seeking tasks, documents and requests are represented by some characterization language. Representations (profiles) are usually made of bags of weighted *terms* (also called *features* in document classification) derived from documents, and allow the computation of similarity between documents and requests (Figure 1.1). In environments involving relevance information, — e.g., routing, classification, or filtering — training documents may be exploited to build better representations of the request, improving effectiveness.

Let us concentrate on the filtering task. The process of constructing or updating topic profiles from training data mainly consists of the following steps:

1. **term selection** (also called *feature selection* in classification environments): Even moderate-sized training data may contain thousands of terms; nevertheless, not all of them are suitable or necessary for representing an information request.

2. **learning of term weights** (or *term weighting* within profiles): Suitable terms differ in their ability to represent the request, thus they have to be weighted accordingly.

The extensive research heritage in retrieval and the close similarity of filtering and retrieval tasks have led researchers to see filtering as an attractive application for techniques that have been developed for retrieval (Belkin and Croft, 1992). As a result, qualitative differences of filtering are usually overlooked.

Current term selection and term weighting techniques have been originally developed in the traditional retrieval context. These techniques mostly consider training data as unordered sets of documents, totally disregarding their time of arrival. To our knowledge, temporal information has not been widely explored in information seeking environments. The most closely related subject is *topic detection and tracking (TDT)*, i.e. the identification of novel events in news streams (Yang et al., 1998; Allan et al., 1998).

In this study, we investigate ways of incorporating temporal information into profile construction by assuming that terms which are distributed uniformly in time are more valuable than others. In Section 3.2, we elaborate on this hypothesis. In Section 3.3, we define a *term occurrence uniformity* measure. Section 3.4 describes the two temporally-dependent term selection methods we have experimented with. Both of them are compared to the baseline of selecting terms with *document frequency thresholding*. Document frequency thresholding has recently proved to be more than an *ad hoc* approach and quite powerful for feature selection in categorization environments (Yang and Pedersen, 1997). The experimental setup, properties of the dataset, and pre-processing are discussed in Section 3.5. We give experimental evidence on how our proposed schemes perform, in Section 3.6.

3.2 The TOU Conjecture

Term selection and term weighting techniques developed for retrieval tasks usually consider document collections as unordered sets. Thus, the arrival time of documents in filtering is totally disregarded when selecting and weighting terms using traditional retrieval techniques. Quoting David Hull from the TREC-7 Filtering Track (Hull, 1998):

> *"...no one has yet explored whether the distribution of a feature over time is related to its usefulness as a discriminator for relevance."*

Changes in the distribution of a feature over time may indicate several things, for example:

- A slow monotonic change in the occurrence rate of a term in relevant documents may indicate a *relevance drift* (Section 2.2), e.g. a slow shift in the focus of the user interest over time, or in the content of relevant documents. In either case this means that arriving relevant documents tend to be different than training data, and this difference is becoming greater over time. The quality of filtering will slowly degrade, unless an adaptive filter responds adequately to these changes.

- A sudden increase of the occurrence rate of a term in relevant documents may indicate a *temporal event*. For instance, *NYC subway bombing* is an event relevant to the topic *terrorism*. Such important events are usually associated with bursts of incoming documents for some period of time. A fast-responding filter, trained for topic *terrorism*, could deceptively be adapted as a result of the very frequent occurrence of terms *NYC* and *subway*, which in general are not characteristic keywords of terrorism.

Drifts in user interest and document content are related in the sense that the idea of relevance changes. We have introduced *locally adaptive systems* in Chapter 2, and we will demonstrate in Chapter 5) how such drifts may be treated by a decay operation. In this study, we assume that requests are stable (no shifts of any kind) but the topic has temporal events.

3.2.1 Terms and Relevance

Let as consider a document stream, e.g. an electronic Newswire issuing, on a daily basis, articles about politics, sports, and entertainment. Let us also assume a user with a certain stable interest, e.g. *football*. With respect to their discriminating power for relevance, terms may be classified as:

- **relevant**: they tend to occur in relevant documents, and are characteristic terms of the topic, in the sense that they are not too ambiguous even when are taken out of context, e.g. *football, goalkeeper, world-cup, Hillsborough* (the Hillsborough disaster), *Pelé*.

- **spurious**: they tend to occur in relevant documents, but are not characteristic terms of the topic. For example, the word *dollars* (think of dollars spent for player transfers) occurs in different contexts as well, e.g. in politics (think of government funds).

- **indifferent**: they tend to occur either too frequently in natural language in general, or too infrequently in relevant documents. Common *function words* (stop-words), e.g. *and, the, must*, belong in the former case. Stop-words have very low semantic content and occur in almost all documents, thus are incapable of characterizing anything in particular (non-discriminating). Conversely, words that occur too *sparsely* to make any significant difference in classification (too discriminating), e.g. *Jenkins* (the goal-keeper of Rising Hope FC) or misspelt words, belong in the latter case.

- **non-relevant**: they do not occur, in general, in relevant documents, and if they do, their occurrence is infrequent and accidental, e.g. *moonshine*.

Let us expand a bit more on relevant terms, but now according to their distributions in time. The temporal classification of topics into *simple periodic*, *uniform*, *periodically clustered*, and *aperiodically clustered*, which we have introduced in Section 2.3, can be applied to relevant terms as well. In this respect, *world-cup* is a periodically clustered term, since every four years and for a period of a few weeks most games are played for the world-cup. *Hillsborough* is an aperiodically clustered term since such disasters are associated with unexpected bursts of relevant documents. We will not go, for now, into a further distinction of terms into uniform or periodic. We will just combine the terms that do not exhibit serious clustering in time (e.g. *football* or *goalkeeper*) under the class of *regularly relevant*, while the rest as *temporally clustered*. Table 3.1 summarizes the relevance classification of terms we have just considered.

2-level classification of terms		**examples** (for topic *football*)
relevant	regular	*football, goalkeeper*
	temporally clustered	*world-cup, Hillsborough, Pelé*
spurious		*dollars*
indifferent	sparse	*Jenkins*
	common function	*and, the, are, must*
non-relevant		*moonshine*

Table 3.1: A classification of terms according to their discriminating power for relevance.

Training data may contain too many terms. Thus, it is not uncommon to end up with thousands of terms in the indexing vocabulary. In the weighting phase, a large number of terms is difficult to handle for learning algorithms. For instance, few neural networks can handle a large number of nodes, and probabilistic models will be computationally intractable unless term independence is assumed. Fortunately, most of the terms can safely be discarded as non-representative for a topic, dramatically reducing the dimensionality of the indexing space.

3.2.2 The Term Temporal Locality Hypothesis

Term selection and weighting schemes that disregard time, make no distinction between regularly and temporally-clustered relevant terms. Should these terms indeed be distinguished and treated differently? We can speculate why taking distributions of term occurrences over time into account may be useful:

> **The Term Temporal Locality Hypothesis:** *Terms occurring frequently over a short period of time, rather than distributed evenly over the whole time-line, indicate a temporally local event and thus they do not have lasting predictive value.*

If this hypothesis is true,

- it can be used as an additional term selection mechanism. The corresponding terms can be removed without a negative impact in filtering effectiveness, but with a desirable benefit for efficiency. In fact, effectiveness may also improve slightly for the same reasons it improves in classification tasks (see Section 3.4) or if these terms happen to be noise terms. Alternatively,

- terms with temporally clustered occurrence characteristics can be down-weighted by learning algorithms, hopefully reducing *classification noise* and gaining effectiveness.

In this study, we investigate this hypothesis in a term selection context. First, we need a tool to distinguish between terms occurring frequently over a short period of time and others.

3.3 A Measure for Term Occurrence Uniformity

Given a stream of documents relevant to a topic, for each occurring term we qualitatively define *term occurrence uniformity* as:

> **Term Occurrence Uniformity (TOU):** *the degree to which the term occurrences are fairly distributed in every possible interval of the stream with respect to interval's length.*

The word "uniformity" here should not be interpreted in a strict probabilistic sense. Probabilistically, a uniform time distribution most probably exhibits a high degree of the fairness we are talking about. But according to our definition, terms which occur in a simple periodic manner will have higher TOU than truly random occurrence patterns (see Figure 2.2).

To measure uniformity according to the definition above, the statistic we have introduced in Section 2.3.1 may be recruited once more. We define the TOU of a term with respect to a topic as

$$U = 1 - V_{\mathrm{RDF}} \ , \tag{3.1}$$

where V_{RDF} is the value of Kuipers' statistic (Equation 2.1) for the occurrence pattern of the term, and RDF is the length of the pattern measured in the number of relevant documents the term occurs in. U takes values in $[0, 1 - \frac{1}{\mathrm{RDF}}]$. and the smaller the U the less fairly a term is distributed according to the definition of TOU.

3.4 Term Selection Methods

The goal of term selection is to reduce the dimensionality of an indexing space without reducing classification accuracy. Given training data for a topic, the terms that do not occur in relevant documents can directly be identified and removed. Traditional stop-listing will eliminate common function words, and light document frequency (DF) thresholding can remove sparse terms. Part-of-speech tagging has also been used for removal of common function words (Rüger, 1998; Arampatzis et al., 2000d) (discussed in Section 7.2.3.1). The remaining spurious and relevant terms may still amount to hundreds or thousands for moderately large relevant training data, however. Automatic *term selection* (or feature selection) methods can remove more of these terms according to training data statistics.

Applying feature selection techniques to text classification tasks has been found not to impair classification accuracy even for reductions up to a factor of ten. In fact, feature selection techniques may slightly improve classification (Lewis, 1992; Yang and Pedersen, 1997; Ragas and Koster, 1998). Possible reasons for these improvements are — despite the fact that less information is actually used — the prevention of over-fitting a classifier into the training data, and the decrease[1] in violations of the feature independence assumption of probabilistic and vector space models.

[1] As the size of a feature set grows, the number of stochastically dependent features grows as well.

(Yang and Pedersen, 1997) performed a comparative evaluation of the most popularly used feature selection methods: document frequency thresholding, expected mutual information, χ^2 statistic, term strength, and information gain. In this study, it turned out that the supposedly *ad hoc* DF thresholding presents a performance comparable to the theoretically justified and best performing schemes like χ^2 and information gain, for term removal up to 90%. The term scores of the latter three methods were found to be strongly correlated, so DF thresholding can be used instead of the others where these are computationally too expensive[2].

3.4.1 DF-thresholding in Filtering

In classification tasks, learning is applied to a single pool of terms which serve to separate documents belonging to different classes. Filtering can be seen as a *binary* classification task where each document has to be classified under one of the two classes: relevant or non-relevant. In this respect, each filtering topic is treated independently of others, therefore it utilizes its own pool of terms. The success of DF thresholding for term selection in non-binary classification tasks demonstrates the importance of terms which occur across classes.

In filtering, however, the two classes are usually too imbalanced: there are many more non-relevant than relevant documents in a stream. DF thresholding on term statistics of the whole stream could hurt topics by eliminating too many of their relevant terms. Therefore, DF thresholding should be applied individually for each topic with respect to the size of its relevant training data. Consequently, the approach of RDF (*relevant document frequency*) thresholding is more suitable than DF in filtering contexts. We define the RDF of a term with respect to a topic as the total number of relevant documents in which the term occurs.

3.4.2 Temporally-dependent Term Selection

We have experimented with two temporally-dependent term selection schemes. The first is based on the *order of arrival (time-order)*, and the second on the actual *time of arrival (time-stamp)* of relevant documents. The approaches are identical when relevant documents arrive with a constant rate of documents per time unit.

The time-stamp uniformity U_{ts} of a term is measured directly with Equation 3.1. All time-stamps are normalized into real numbers in the interval $[0, 1]$, where the first document of the training part of a stream is located at 0, and the first document of the test part at 1. To measure time-order uniformity U_{to} with the same Equation, we assume a discrete time-line: R relevant documents are assumed to have arrived at the normalized time-points i/R, $i = 1, \ldots, R$. In both cases, terms inherit the lists of normalized time-points of the documents they occur in.

[2] The success of DF thresholding for term selection (i.e. rank all candidate terms according to their document frequency and select the most frequent ones) may sound counterintuitive. At least, the procedure is bound to fail at very aggressive cutoffs (e.g. selecting only the top-10 terms) especially if stop-list is not used, since it will mostly select common function words. (Yang and Pedersen, 1997) used a stop-list. Furthermore, they have not used dangerously aggressive cutoffs. Their most aggressive term removal reported for Reuters was 98% which resulted in 321 terms.

In a similar manner as in DF thresholding, we have experimented with ranking the candidate terms with our proposed schemes and then applying different cutoffs keeping only the high scoring terms for training. We have compared both U_{to} and U_{ts} thresholding to RDF thresholding. Before we report the experiments, it will be useful to see what the relation between RDF, U_{to}, and U_{ts} is. Equation 3.1 has the following properties:

1. $\mathrm{RDF} = 1 \Rightarrow U = 0$ (easily deduced from Figure 2.3), and
2. $\lim_{\mathrm{RDF}\to\infty} U = 1$ (generally not true for all time distributions, but provable in our context since there is always some time distance between consecutive relevant documents).

The first property will conveniently score the terms that occur only once (too sparse) at the bottom of the rank. The latter property implies that the *expected value* of U is directly related to RDF. This effect may be desirable for small RDF in order to devalue more sparse terms, but it also indicates a certain bias of U to RDF.

Figure 3.1 gives values of U for 10,000 randomly generated term occurrence patterns with up to 200 occurrences. The plot at the top corresponds to U_{to}. In this case, for R relevant documents, terms can occur only at points i/R, $i = 1, \ldots, R$. Obviously, the correlation between RDF and U becomes stronger as RDF gets close to R (the spread of data in the figure diminishes). The plot at the bottom corresponds to U_{ts}. In principle, the time-line is now continuous, since relevant documents can arrive as temporally close to each other as one may think. The rate at which the correlation between RDF and U_{ts} becomes stronger, as RDF tends to R, is now lower than the discrete case (the spread of the data does not become considerably thinner for large RDF, but it becomes asymptotically thin only when $\mathrm{RDF} \to \infty$). In practice, the arrival rate of documents is always bounded due to processing power and network speed limitations. Thus, this correlation will be somewhat stronger.

In general, it could be proved that any term occurrence uniformity measure is correlated in some way to relevant document frequency, and the correlation becomes stronger as relevant document frequency becomes larger. Especially U_{to} tends to produce the same rank of terms as RDF when $\mathrm{RDF}/R \to 1$ (this usually but not necessarily happens at the top of the rank). Therefore, RDF and U_{to} are expected to result in comparable effectiveness at aggressive cut-offs, something that is not guaranteed for U_{ts}.

3.5 Experimental Setup

The experimental system is based on the vector space model with a dot-product similarity function (Section A.1), terms are weighted in a *ltc* fashion (Section A.2.1), and classifiers are constructed automatically using Rocchio's relevance feedback method (Section A.4). We used the original Rocchio formula, that is, $\alpha = 0$ and $\beta = \gamma$.

In order to abstract away from the threshold selection problem, we evaluate in a routing setting: We allow the system to return a traditional *ranked list* of documents for every profile: most relevant first, least relevant last. Thus, evaluation is done with 11-point interpolated average precision (Section A.5.2).

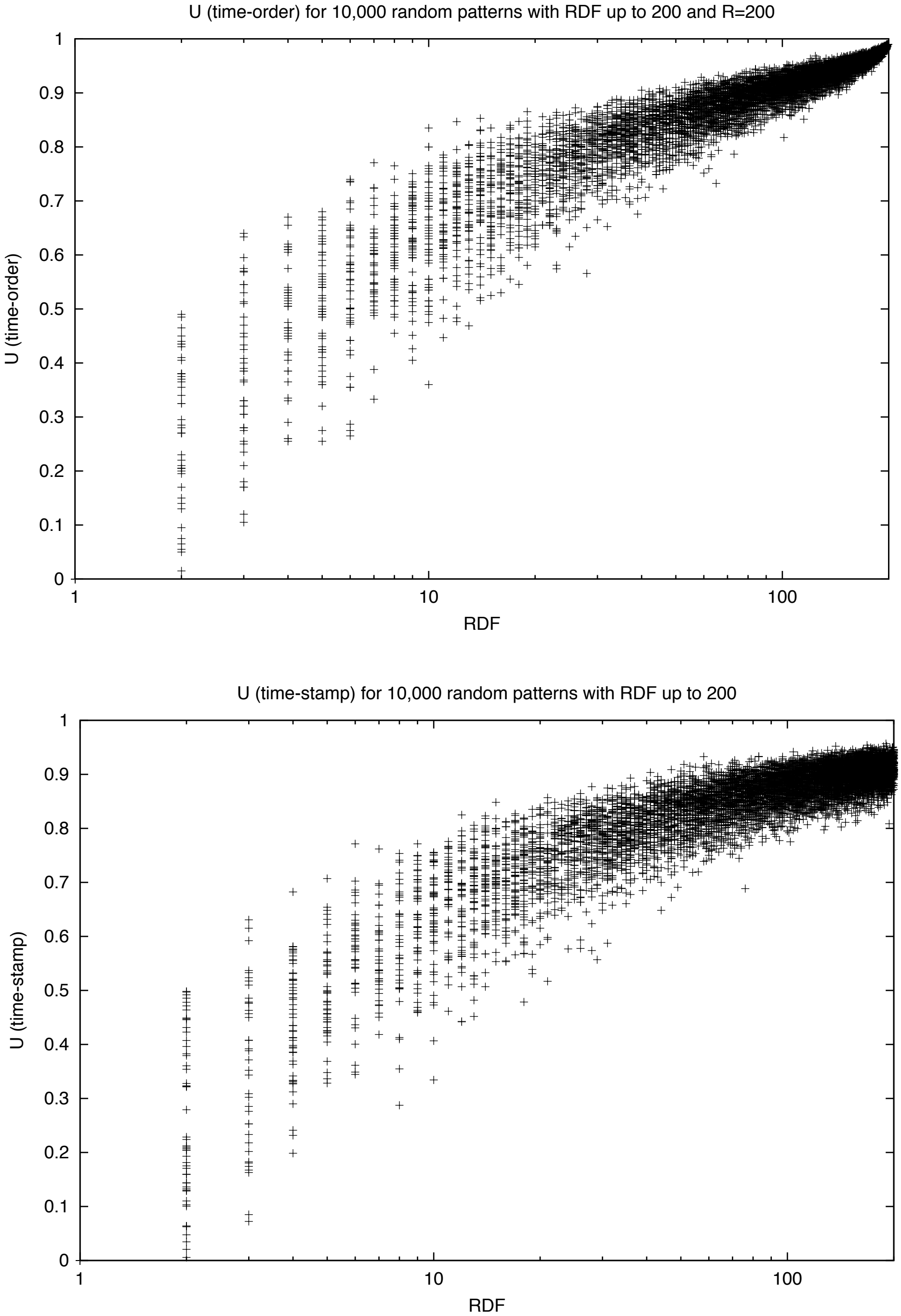


Figure 3.1: Correlations between RDF and U for random occurrence patterns.

As a dataset we use the Reuters-21578 text categorization test collection (Section A.6.1). We use only the topics which have at least 100 relevant training documents (16 topics in total). All training documents which do not belong to any of these topics were removed from the training set. Table 2.1 shows the 16 largest topics and their corresponding relevant document counts in the training and test stream. The training stream covers a period of 40.4 days.

For these experiments, we use single-word terms. The pre-processing phase is performed in four stages: tokenization, part-of-speech (POS) tagging, removal of common function words, and morphological normalization of the remaining words.

Tokenization consists of detection of sentence boundaries, followed by division of sentences into words. Detection of sentence boundaries is necessary since we use a POS-tagger. Brill's rule-based tagger[3] (Brill, 1994) was employed to obtain POS information for the words of the dataset. We use a *POS stop-list* to remove all common function words; we remove all words except: nouns, adjectives, verb-forms, and adverbs. Morphological normalization of the remaining words is performed by means of *lemmatization* (which can be seen as a form of POS-directed stemming), using WordNet's v1.6 (Miller, 1995) morphology library functions[4].

3.6 Results and Discussion

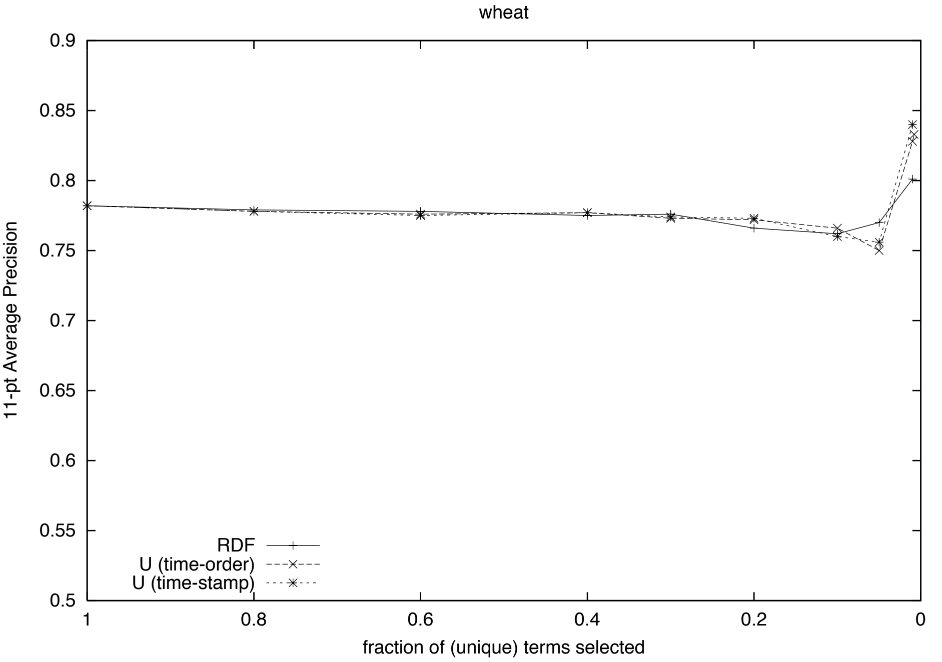
Figures 3.2, 3.3, and 3.4, depict some representative experimental results; the curves for the 10 remaining topics can be found in (Arampatzis et al., 2000c). For each topic, all potential terms (the ones which occur in relevant training documents) are ranked by each of the term selection methods under investigation. The 11-point average precision is plotted as a function of the fraction of terms selected, from 1 (all terms) down to 0.01 (99% of all terms are eliminated). In fact, for each term selection method and topic, cutoffs are applied only down to the lowest point that does not result in empty relevant training documents. All terms with single occurrences are eliminated in advance.

To begin with, our term selection results agree with previous research (Lewis, 1992; Yang and Pedersen, 1997; Ragas and Koster, 1998): most of the terms in classification environments can be eliminated without impairing classification effectiveness (as this measured by average precision); even slightly improving it for some topics.

Average precision increases drastically for topics *wheat*, *sugar*, and *coffee* for aggressive cutoffs for all term selection methods (Figure 3.2). This result seems counterintuitive at first glance. After further investigation it is found that these topics have words which occur in almost all of their relevant documents (unique identifiers). These words unsurprisingly are *wheat*, *sugar* and *coffee* and occur in 97%, 96%, and 100% of the relevant documents of the respective topics. A unique identifier together with a few other terms

[3] Eric Brill's tagger V1.14 and a description are available by anonymous ftp from: `ftp://ftp.cs.jhu.edu/pub/brill` in the `Programs` and `Papers` directories.

[4] Specifically, we called the `morphstr()` function which tries to find the base-form (lemma) of a word or collocation, given its part-of-speech. WordNet is created by Cognitive Science Laboratory, Princeton University, 221 Nassau St., Princeton, NJ 08542. It is available for anonymous ftp from `clarity.Princeton.edu` and `ftp.ims.uni-Stuttgart.de`.



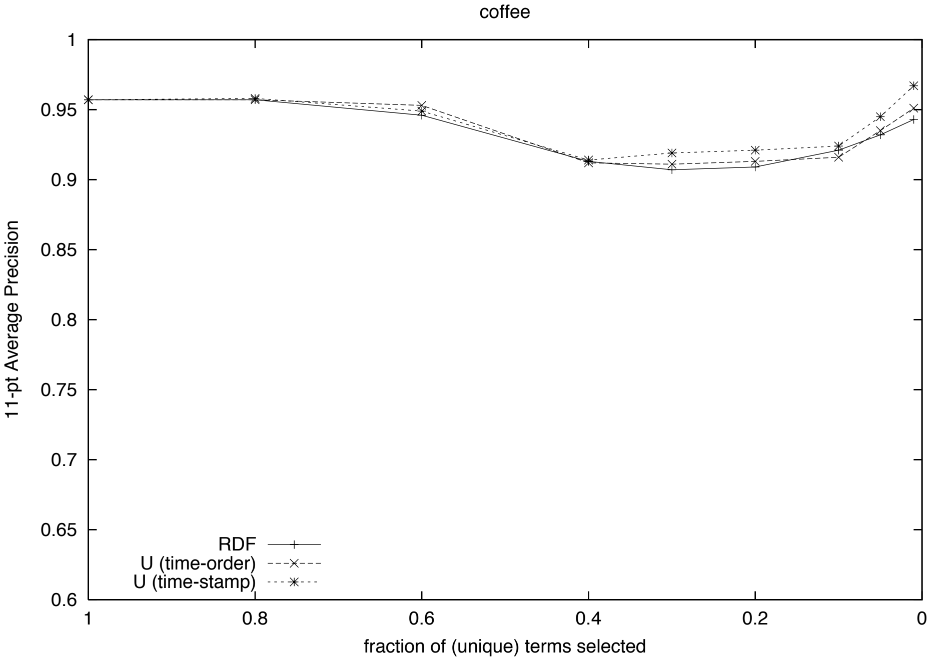


Figure 3.2: Topics with unique identifiers. TOU works better than RDF.

have proved sufficient for achieving the best results for those topics. In those cases, average precision is maximized for classifiers with 9–12 words, while a larger number of words is likely to introduce noise rather than improve effectiveness. It is important to note that for those topics our methods have performed better than RDF at aggressive cutoffs, suggesting that they select more discriminating words to accompany the unique identifiers in classifiers.

In a comparison between RDF, U_{to}, and U_{ts}, all methods present a comparable performance (< 0.05 points of average precision) for reductions up to 90% for most topics (Figure 3.3). At more aggressive cutoffs, however, RDF seems to perform generally better than uniformity-based term selection (Figure 3.4). Nevertheless, even here the difference in average precision is in general less than 0.10 points. Thus, all methods seem to hold up comparably at aggressive cutoffs.

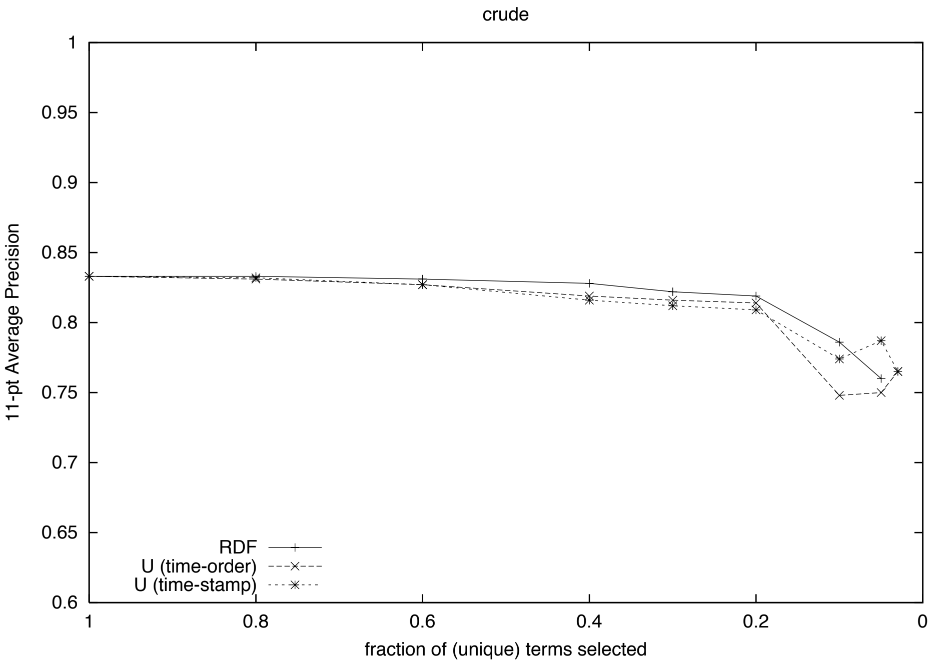
The fact that U_{to} allows, in general, more aggressive cutoffs than U_{ts} (meaning that it does not result soon in empty relevant documents) is a consequence of its stronger correlation to RDF. The correlations of U_{to} and U_{ts} to RDF for Reuters are given in Figure 3.5. We normalized RDF as RDF$/R$ per topic, so the plots make more sense when U_{to} values for all topics are plotted together. The obvious upper bound of U_{ts} values is partly a consequence of the lack of documents arriving in weekends, as we mentioned earlier in Section 2.3.2. It is also because of the continuous time-line considered, a consideration which produces in general lower values than U_{to}.

Although both TOU methods present a correlation to RDF as $R \rightarrow$ RDF, this correlation rather diminishes for frequency characteristics with which most of the terms occur, e.g. for RDF$/R < 0.1$. This observation suggests that our methods are indeed novel, since they throw away quite different sets of terms than the RDF method, for moderately aggressive cutoffs. Nevertheless, the fact that we see no improvements in performance at these cutoffs implies that we have been looking for local events where their recognition is not of great importance for classification, e.g. in the Reuters collection.

On the one hand, the fact that the TOU term selection methods show a performance comparable to RDF for reductions up to 90% appears promising, since document frequency thresholding is known to be a powerful method for term selection. On the other hand, if most of the terms are to be thrown away, what matters most for a term selection method is to achieve high accuracy at very aggressive cutoffs. At those cutoffs, while there is no sharper decrease in effectiveness with our methods, document frequency thresholding seems more reliable.

3.7 Summary

We have taken up the challenge by David Hull (Hull, 1998) and investigated the use of time distributions in retrieval environments with temporally-dependent data. We have introduced *term occurrence uniformity (TOU)* as a novel term selection method with a performance comparable to document frequency thresholding. We regard this result as promising, since document frequency thresholding is known to be more than just an *ad hoc* approach for term selection, and quite powerful in text categorization environments (Yang and Pedersen, 1997). The hypothesis of temporal locality of terms has been neither



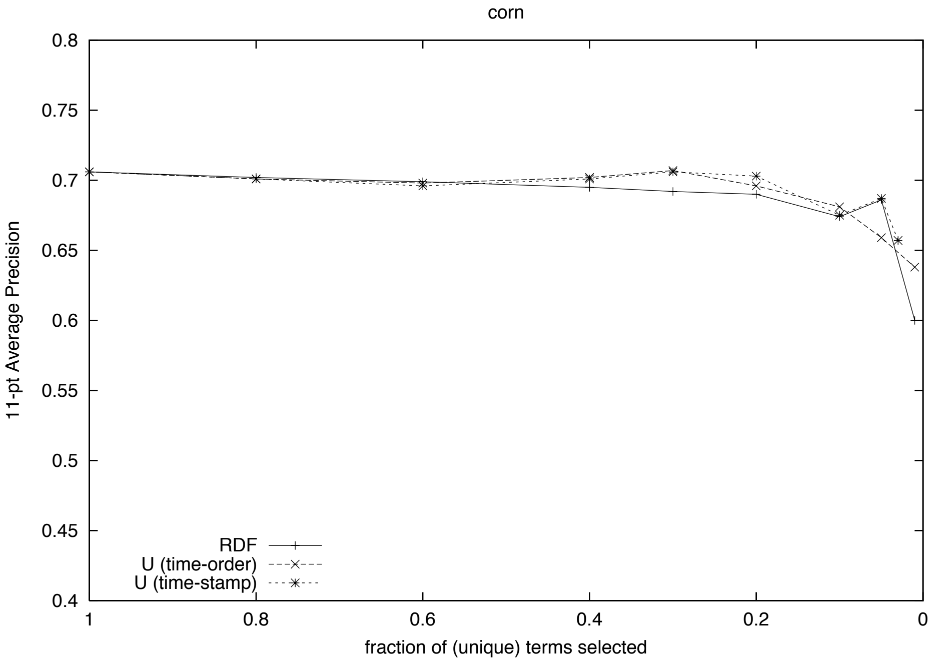
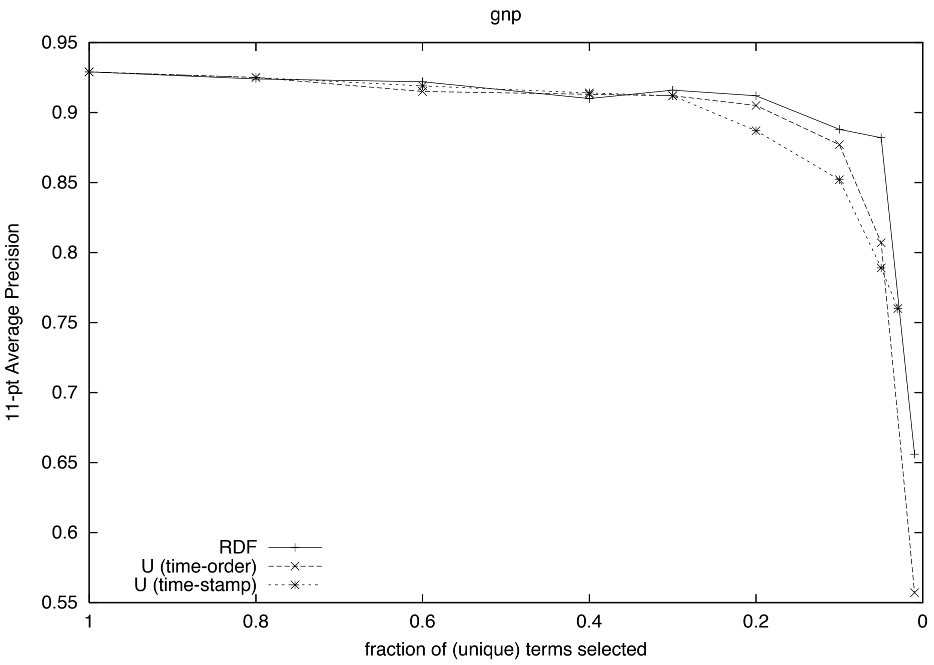


Figure 3.3: TOU is comparable to RDF.



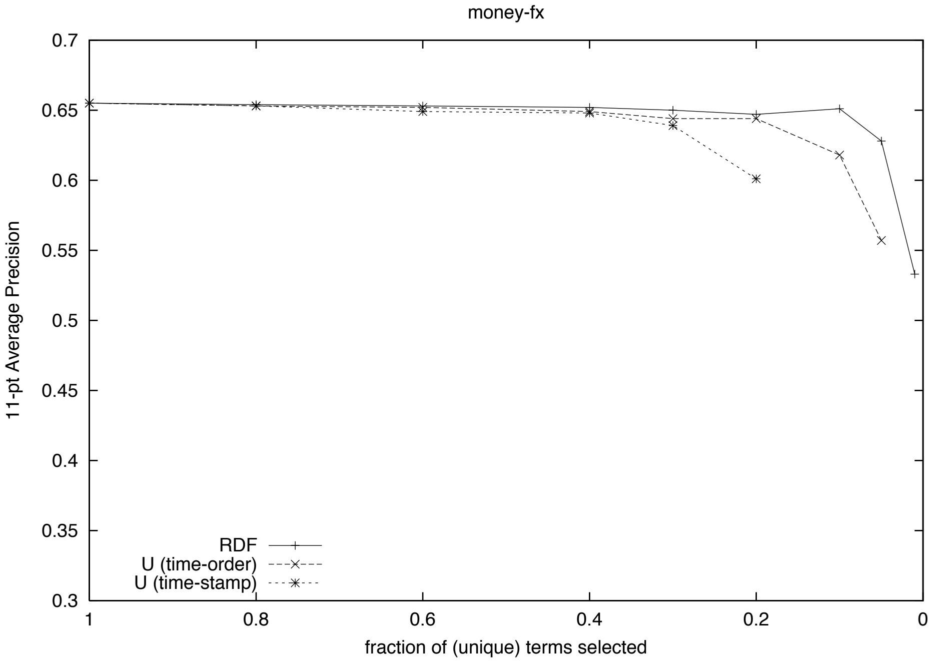
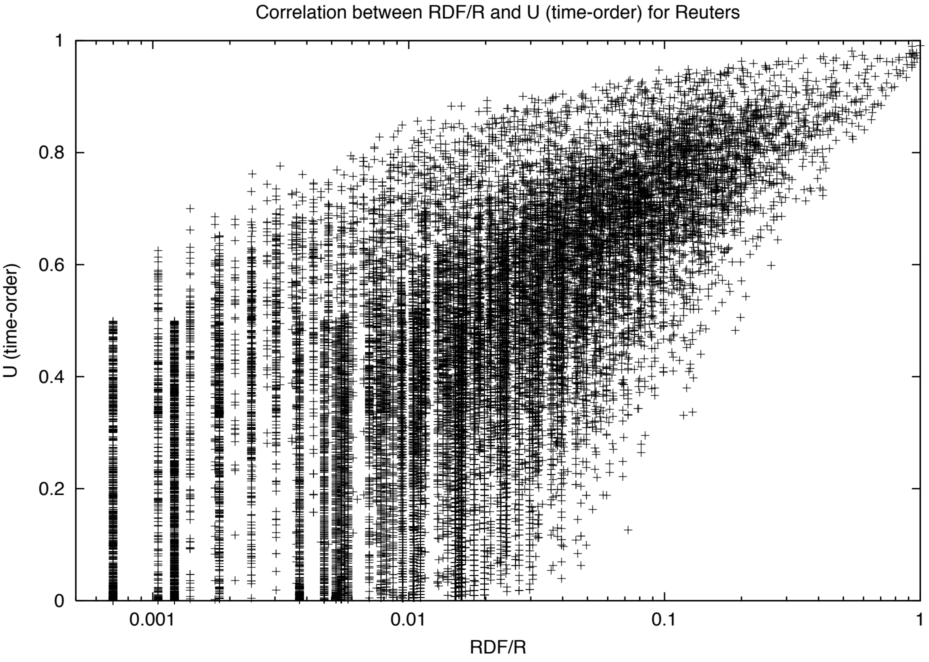


Figure 3.4: RDF performs better than TOU.



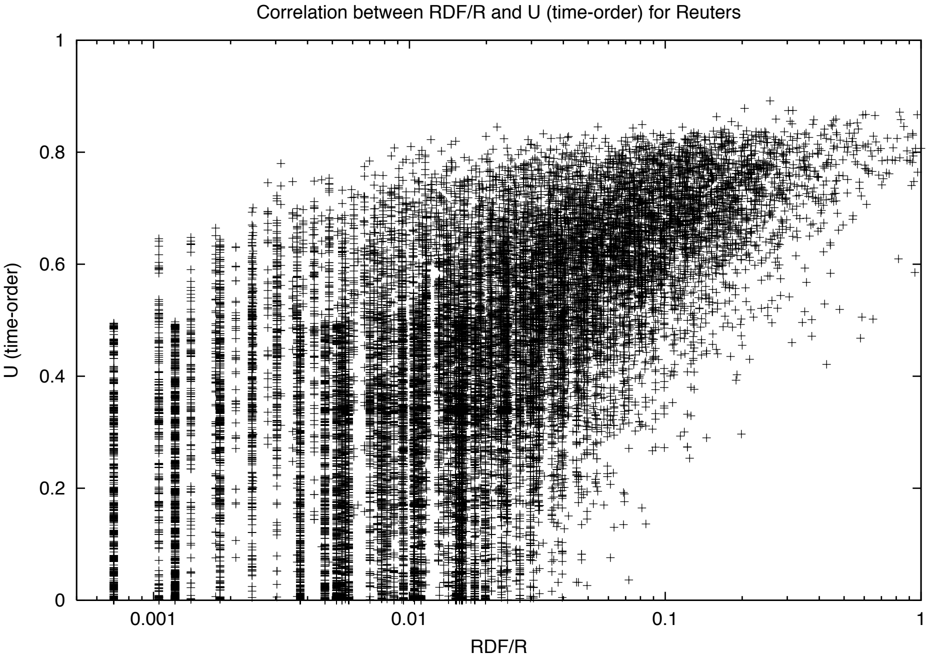


Figure 3.5: Correlations between RDF and U_{to} (top), U_{ts} (bottom), in Reuters.

proved nor disproved, since our results are rather inconclusive. The subject indeed merits deeper theoretical and empirical investigation.

To keep the ball rolling, we will summarize what we believe has influenced our approach to proving the hypothesis, and suggest directions for further research:

1. We believe that the Reuters-21578 collection is improper for this kind of research. The training period is short, covering slightly over 40 days, which gives little scope for temporally local events and non-uniformity. We will have to repeat the test with material collected over longer periods of time, and larger numbers of relevant training documents. Unfortunately, the availability of such test data still remains a big issue.

2. The approach taken has been a brute-force one; candidate features were ranked simply according to their uniformity. A wise integration of a TOU method and some other powerful time-disregarding term selection method may combine the benefits of both approaches. We believe that U_{ts} is a better candidate for such an integration, since it better reflects the actual event identification. Temporal events should be taken into account only when they introduce serious clustering of data in time. If this is not the case, the new method should turn into a time-disregarding one, since uniformity measures are (weakly but) correlated to other term selection methods.

3. While our intention is to develop temporally-dependent term selection and weighting schemes for filtering, we have tested our approach in a rather static situation, namely document routing, with clearly defined training and test phases. A real-world filtering task is usually an adaptive process. Adaptive filtering is an especially sensitive task. Therefore, the application of such temporally-dependent term selection and term weighting schemes in adaptive filtering is expected to show larger variations in effectiveness.

4. The temporal classification of topics we have introduced in Section 2.3 has not been taken into account. The distribution of a topic in time can provide useful information. For instance, *terrorism* is an *event-driven* (aperiodically clustered) topic, in the sense that documents about terrorism occur mostly when a related event happens, e.g. a NYC subway bombing. Compare this to *football* which is usually a rather *event-irrespective* (periodically clustered) topic. Football developments are reported on a regular basis, irrespective from whether something really important has happened (unless some disaster occurs). The distribution of a topic in time reflects on the time distributions of its terms, a fact that must be taken into account.

At any rate, the issue of using time distributions in retrieval tasks is not settled. We are, however, convinced that Information Retrieval in general could benefit by taking the effect of time into account.

Chapter 4

Threshold Optimization

4.1 Introduction

Traditional retrieval systems display documents in decreasing order of their *scores* with respect to a request. A score may correspond to the probability of relevance of the document, or to some other similarity measure. A user is supposed to go down such a ranked list of documents, and stop at some point determined by the satisfaction (or dissatisfaction) of his/her request. In some retrieval applications, however, rankings are not enough.

In *binary classification* tasks, e.g. document filtering, a decision should be made for every document as to whether it belongs to a given class or not. If a system is supposed to operate over long periods of time, the interaction between the system and users should be minimized due to cost factors. Decisions such as where to "cut" a ranked list have to be made automatically by the system. In some cases, decisions are required to be made as soon as a document arrives, therefore ranked lists are not even possible. These issues suggest the *thresholding* of document scores.

The degree of satisfaction or dissatisfaction of a user may be expressed by an *effectiveness measure*, and the goal of a system is to *optimize* this measure. Thresholding strongly affects effectiveness, and there is no single threshold which optimizes all effectiveness measures. As an example consider two users: the first user values every relevant document as 1 unit of currency, the second user as 10 units, while a non-relevant document costs both users 1 unit. As we will see later, such gain–cost considerations are best captured by *linear utility functions*. Assuming that a ranked list has more and more non-relevant documents at lower ranks, the gain of the first user will peak at a higher rank than that of the second. Thus, the corresponding *optimal thresholds* are different.

A classification system operating over long periods of time may accumulate history, e.g., documents and maybe relevance judgments. History can be used to alter the classification model, and thus make better predictions in the future. Systems that alter the classification model in response to the history are called *adaptive*. Adaptive systems should be able to perform updates in a limited number of calculations and memory. These practical considerations suggest that only a portion of the history should be retained, and algorithms ought to be implemented *incrementally*.

This chapter is based on our work previously reported in (Arampatzis and van Hameren, 2001). In Section 4.2, we review the most important related approaches to threshold optimization. In Section 4.3, we introduce the *score-distributional (S-D) threshold optimization* method, capable of optimizing any effectiveness measure defined in terms of the traditional contingency table. The method is based on score distributions.

In Section 4.4, we provide a model for estimating score distributions and demonstrate its accuracy in describing empirical data. Our work in modeling score distributions is useful beyond threshold optimization problems. It can be applied to other retrieval environments that may require such a modeling, e.g., distributed retrieval (Baumgarten, 1999), or topic detection and tracking (Spitters and Kraaij, 2000). Nevertheless, our model — although incremental — can be computationally rather heavy.

In Section 4.5, we set out to investigate practical solutions. We suggest practical approximations and discuss adaptivity, threshold initialization, and incrementality issues. In Section 4.5.3, we give a practical method for optimizing linear utility functions. An early version has been tested in the context of the TREC-9 filtering task and found to be very effective (Arampatzis et al., 2000a); we will report the empirical evaluation in Chapter 5.

4.2 Optimizing Thresholds

Let us assume that a set of n documents has been judged by a user, and that r of them have been found relevant to a certain request. Then, the same set of documents is given to a classification system which makes a decision for each document whether to retrieve it or not. All possible four combinations of the user's judgments and the system's decisions can be summarized (quite traditionally) in the contingency Table 4.1. The variables R_+, N_+, R_-, N_-, refer to the number of documents in each category.

system's	**user's judgment**	
decision	relevant	non-relevant
retrieved	R_+	N_+
non-retrieved	R_-	N_-
total	r	$n-r$

Table 4.1: The traditional contingency table.

Effectiveness measures in retrieval tasks are usually defined as functions of the above four variables. Through the years, a wide range of effectiveness measures have been defined, e.g., precision, recall, the F measure, error rate, and utility, just to name a few popular ones.

4.2.1 The Probability Thresholding Principle

From the point of view of optimizing measures, D. Lewis in (Lewis, 1995a) formulates the *probability thresholding principle* (PTP):

> *"For a given effectiveness measure, there exists a threshold p,* $0 \leq p \leq 1$*, such that for any set of items, if all and only those items with probability of class membership greater than p are assigned to the class, the expected effectiveness of the classification will be the best possible for that set of items."*

The PTP is a strengthening of the *probability ranking principle* (Robertson, 1977) to address the limitations of the latter in classification environments.

The PTP creates two categories of effectiveness measures: measures for which the PTP applies, and measures for which it does not. For the former measures, optimizing a threshold is *theoretically* trivial (we will see the practical difficulties later). A threshold on probability of relevance can be set once, and the system is guaranteed to exhibit optimal effectiveness in the future, no matter what the distribution of probabilities of relevance for documents is.

As an example, let us consider the family $U_{(\lambda_1,\lambda_2,\lambda_3,\lambda_4)}$ of linear utility functions:

$$U_{(\lambda_1,\lambda_2,\lambda_3,\lambda_4)} = \lambda_1 R_+ + \lambda_2 N_+ + \lambda_3 R_- + \lambda_4 N_- \ , \tag{4.1}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ denote the gain or cost associated with each document that falls under the corresponding category. The optimal probability threshold associated with any of those functions has been shown in (Duda and Hart, 1973) to be:

$$p = \frac{\lambda_2 - \lambda_4}{(\lambda_3 - \lambda_1) + (\lambda_2 - \lambda_4)} = \frac{1}{1 + \lambda} \ , \tag{4.2}$$

where

$$\lambda = \frac{\lambda_3 - \lambda_1}{\lambda_2 - \lambda_4} \ . \tag{4.3}$$

Since the optimal threshold depends only on the measure, the PTP holds.

Practically, such probabilistic thresholds are difficult to apply. The main reason is that even probabilistic retrieval models do not obtain the actual probabilities of relevance for documents. Traditional probabilistic models make extensive use of *order-preserving* transformations (some of which are difficult to reverse) of probabilities of relevance. Any such transformation does not affect ranked retrieval, but makes formulae like Equation 4.2 practically useless, unless a way is found to reverse the transformations. A transformation reversal strategy has been adopted by the Okapi probabilistic system with rather successful results (Robertson and Walker, 2000).

For non-probabilistic retrieval models, however, how to turn a similarity score into a probability of relevance is still a fair question. In any case, optimizing measures for which the PTP does not hold (e.g., the F measure) require other considerations. A method based on *score distributions*, irrespective of what a score is, would be more general and valid for any measure or retrieval model.

4.2.2 The Straightforward Method

There exists a distributional procedure, which we will call the *straightforward empirical* method, that guarantees to find an optimal threshold on training data. It consists of the following steps:

- calculate the scores of all training documents,
- rank them,
- calculate the effectiveness measure at every position of the rank,
- go down the rank and find the position where the effectiveness measure is optimal,
- set the threshold somewhere between the score that corresponds to the position above and the next one.

The technique implicitly considers the density of relevant to the non-relevant documents and the spread of their scores. It has been applied many times before and, given sufficient training data, works well (see e.g. (Schapire et al., 1998)).

Although the straightforward empirical method seems like a perfect choice for optimizing thresholds in classification tasks, its drawbacks become apparent when adaptivity is required. Firstly, there is no known way to implement it incrementally. The scores of all accumulated training documents have to be re-calculated after every query update, therefore document buffers are required. The fixed memory model requirement of practical systems means that buffers should be of limited size, thus some documents have to be discarded as the history grows. This may have a negative impact on the estimation accuracy, especially when the convergence of classifiers is more important than responsiveness[1]. Moreover, the straightforward method gives absolutely no prediction of where the optimal threshold may be, when there is no relevance information.

Our proposed S-D method has the following advantages over the above empirical technique.

1. It allows for better incrementality, retaining accuracy. Most of the quantities it needs for the estimation can be updated incrementally when new data become available.
2. It can give predictions of where the optimal threshold may be, even when there is sparse relevance information.
3. It uses the statistical properties of the scores rather than the actual values. Therefore, the estimation of the optimal threshold may generalize better to unseen documents.

[1] Responsiveness of classifiers is required when relevance drifts exist. In such cases, old training data may be discarded more safely, since their relevance judgment was valid at the time it was generated and may not correspond to now. Such considerations can be found in Chapter 2.

4.3 The S-D Threshold Optimization

The S-D threshold optimization method can be applied for any effectiveness measure of the form $M(R_+, N_+, R_-, N_-)$, i.e. M is any function of the variables of the contingency Table 4.1. The optimization is based on the score distributions of relevant and non-relevant documents. It takes into account not only the means of these distributions but also the spreads of the data and the relative density of relevant to non-relevant documents in a stream.

Let us assume that the scores of relevant documents are distributed with a probability density function $P_{\mathrm{r}}(x)$. Then, the quantity $rP_{\mathrm{r}}(x)\,dx$ gives the number of relevant documents with scores in the range $[x, x+dx)$. The number of relevant documents which score above a threshold θ is

$$R_+(\theta) = r \int_{\theta}^{+\infty} P_{\mathrm{r}}(x)\,dx \ . \tag{4.4}$$

The number of non-relevant documents with scores above θ is similarly defined as

$$N_+(\theta) = (n-r) \int_{\theta}^{+\infty} P_{\mathrm{nr}}(x)\,dx \ , \tag{4.5}$$

where $P_{\mathrm{nr}}(x)$ the probability density function of the score distribution of non-relevant documents. The numbers of relevant non-retrieved and non-relevant non-retrieved documents for θ are given respectively by

$$R_-(\theta) \ = \ r \int_{-\infty}^{\theta} P_{\mathrm{r}}(x)\,dx \ , \tag{4.6}$$

$$N_-(\theta) \ = \ (n-r) \int_{-\infty}^{\theta} P_{\mathrm{nr}}(x)\,dx \ . \tag{4.7}$$

Using the last four equations, M can be written as a function of θ:

$$M\left(R_+(\theta), N_+(\theta), R_-(\theta), N_-(\theta)\right) \ . \tag{4.8}$$

Optimizing M means either maximizing or minimizing it (depending on whether larger M means better effectiveness or the other way around), therefore the optimal threshold is a solution of

$$\frac{dM\left(R_+(\theta), N_+(\theta), R_-(\theta), N_-(\theta)\right)}{d\theta} = 0 \ . \tag{4.9}$$

In order to solve this equation for a given M, we first need to define the probability densities $P_{\mathrm{r}}(x)$ and $P_{\mathrm{nr}}(x)$ of the score distributions. We will model these distributions in Section 4.4.

In most cases, Equation 4.9 does not have analytical solutions because of the integrals involved, so it has to be solved numerically. For linear measures, however, it simplifies greatly since the integrals cancel out with the derivative of the measure. For example, for the family of linear utility functions, the derivative of the general function of Equation 4.1 becomes

$$\frac{dU_{(\lambda_1,\lambda_2,\lambda_3,\lambda_4)}(\theta)}{d\theta} = -\lambda_1 r P_{\mathrm{r}}(\theta) - \lambda_2 (n-r) P_{\mathrm{nr}}(\theta) + \lambda_3 r P_{\mathrm{r}}(\theta) + \lambda_4 (n-r) P_{\mathrm{nr}}(\theta)\ . \tag{4.10}$$

By setting this equal to zero, after a few routine calculations it leads to

$$\lambda \rho P_{\mathrm{r}}(\theta) = P_{\mathrm{nr}}(\theta)\ , \tag{4.11}$$

where λ is given by Equation 4.3, and

$$\rho = \frac{r}{n-r} \tag{4.12}$$

is the *relative density* of relevant to the non-relevant documents.

The probability $P(\mathrm{rel}|s)$ of a document with score s to be relevant may be expressed as

$$P(\mathrm{rel}|s) = \frac{r P_{\mathrm{r}}(s)}{r P_{\mathrm{r}}(s) + (n-r) P_{\mathrm{nr}}(s)}\ . \tag{4.13}$$

The probability of relevance at $s = \theta$ can be calculated by using Equation 4.11 on 4.13. The result is $P(\mathrm{rel}|\theta) = \frac{1}{1+\lambda}$, i.e. the same as Equation 4.2. Obviously, our method may be used, via Equation 4.13, to reverse scores into probabilities of relevance, however, we do not see the need to do that since we can calculate the optimal threshold in the first place.

4.4 Score Distributions

(Baumgarten, 1999) has modeled score distributions, using the mean and deviation of the data, with a *gamma distribution* shifted by the minimum score. The motivation for using a gamma distribution has been empirical, but the approach has worked out well. We will instead set out to build a theoretical model from scratch.

Let us represent a query by an m-tuple $\boldsymbol{q} = [q_1, \ldots, q_m]$, where q_i is a value that corresponds to the term i. A document is represented similarly, using the same set of terms, as $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_m]$. The values of the terms in documents depend on a weighting scheme W. Subsequently, $\boldsymbol{q}$ and W together determine the structure of the document space. We will specify W only qualitatively such as: the larger the similarity of a document to the query, the larger the document score defined as the linear function of document weights:

$$\langle \boldsymbol{q}, \boldsymbol{\omega} \rangle = \sum_i q_i \omega_i\ . \tag{4.14}$$

Represented as m-tuples, documents and queries are obviously points in $I\!\!R^m$.

Our aim is to calculate the distribution of the scores of a class C of documents. Since the score of a document is a linear combination of its components, the score distribution can be derived from the distribution of the documents in $I\!R^m$. This distribution can be represented by a probability measure P_m on $I\!R^m$. For every *convex subset* $A \subset I\!R^m$, the number $\mathsf{P}_m(A)$ gives the fraction of documents from C for which their m-tuples are in A[2]. Although a real-life set of documents is countable, we represent it by the *continuous* space $I\!R^m$. The large number of different documents makes this a reasonable approximation.

Of course, the distribution of documents does not have to be smooth in $I\!R^m$, and all documents are restricted to a hyper-surface in $I\!R^m$ of lower dimension than m, say $m-1$. Strictly speaking, we should then define a measure P_{m-1} on this (curved) lower dimensional space. We, however, prefer to formulate everything in $I\!R^m$, and to put possible constraints in P_m with the help of *Dirac δ distributions*. For example, if all documents happen to be distributed on a hyper-sphere in $I\!R^m$ with center $[0, 0, \dots, 0]$ and radius R, then

$$\mathsf{P}_m(d\boldsymbol{\omega}) = P(\boldsymbol{\omega})\delta(\|\boldsymbol{\omega}\| - R)\, d\boldsymbol{\omega}\ , \tag{4.15}$$

where P is a positive function on $I\!R^m$ such that

$$\int_{I\!R^m} P(\boldsymbol{\omega})\delta(\|\boldsymbol{\omega}\| - R)\, d\boldsymbol{\omega} = 1\ . \tag{4.16}$$

The δ distribution restricts the measure to be non-zero only for documents that have lengths equal to R.

Let us denote $[\boldsymbol{\alpha}, \boldsymbol{\beta}) = [\alpha_1, \beta_1) \times [\alpha_2, \beta_2) \times \cdots \times [\alpha_m, \beta_m)$ and $\mathsf{P}_m(d\boldsymbol{\omega}) = \mathsf{P}_m(\,[\boldsymbol{\omega}, \boldsymbol{\omega} + d\boldsymbol{\omega})\,)$. Given P_m, the characteristic function ϕ of the score distribution is given by

$$\phi(t) = \mathsf{E}(\, e^{\imath t\langle \boldsymbol{q}, \boldsymbol{\omega}\rangle}\,) = \int_{I\!R^m} e^{\imath t\langle \boldsymbol{q}, \boldsymbol{\omega}\rangle}\, \mathsf{P}_m(d\boldsymbol{\omega})\ , \tag{4.17}$$

and the probability density of the scores of class C is given by the *Fourier transform* of ϕ (Laha and Rohatgi, 1979):

$$P_C(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{-\imath xt} \phi(t)\, dt\ . \tag{4.18}$$

In the formulation above, the components ω_i of the documents can be considered random variables, and the score is a linear combination of these random variables

$$S_m = \sum_{i=1}^{m} X_i\ , \qquad X_i = q_i \omega_i\ . \tag{4.19}$$

[2] The convexity of A is a fair requirement. Suppose you do not demand A to be convex, for example take A to be such that it consists of tiny balls around the points of the documents, connected by very narrow tubes. Then, P_m will look like a collection of delta peaks at the points of the documents. In order to smooth these peaks out and get a nice continuum limit, the convexity of the subspaces A is required.

We will make the (common) assumption that

Assumption 1 *the components ω_i of documents are distributed independently.*

For the measure P_m, this means that it factorizes over the components of $I\!R^m$, i.e., there are m one-dimensional measures p_i so that we can write

$$\mathsf{P}_m(d\boldsymbol{\omega}) = \prod_{i=1}^{m} \mathsf{p}_i(d\omega_i) \ . \tag{4.20}$$

As a result of this and the linearity of the score as a function of document components, the characteristic function can be written as a product of characteristic functions of the components:

$$\phi(t) = \prod_{i=1}^{m} \phi_i(q_i t) \ , \quad \phi_i(q_i t) := \int_{-\infty}^{\infty} e^{\imath t q_i \omega_i} \, \mathsf{p}_i(d\omega_i) \ . \tag{4.21}$$

In order to construct the one-dimensional measures, we observe that weighting schemes are usually such, that if a term does not appear at all in a document, then this term gets weight zero. We relate the probability of term i to appear in a document directly to its document frequency across class C by defining

$$\varepsilon_i := \frac{\text{number documents in } C \text{ containing term } i}{\text{total number of documents in } C} \ , \tag{4.22}$$

and we call it the *term probability* (TP). Consequently, the measure p_i will have the form

$$\mathsf{p}_i(\,\omega_i \leq x\,) = (1-\varepsilon_i)\vartheta(x) + \varepsilon_i F_i(x) \ , \tag{4.23}$$

where ϑ is the step function, and F_i is some probability distribution function (PDF) which depends on W. F_i is the PDF that corresponds to the probability density of the weights of term i for the documents it occurs in. In the simplest case of binary weighted document terms, $F_i(x) = \vartheta(x-1)$, $\forall i$. In general, the F_i functions can be derived directly from the W being used, or estimated empirically from a dataset.

So far, we have built a model for the score distribution of a class C of documents. The model is capable of calculating the distribution from the term probabilities and the query. The only assumption we have made is that of the independence of term occurrences. We have left the form of functions F_i open; these should be defined according to the W used.

Turning to the independence assumption, our model will more likely work better when there are less violations. This suggest a small number of dimensions m, or that the model should be used for document classes which have a small number of matches with the query, e.g. the non-relevant documents. Dependencies blow up the scores. Our model, however, allows us to take the dependencies indirectly into account, through the functions F_i; these can be adjusted accordingly to compensate for the score blow-ups, as we will see later.

4.4.1 Gaussian Limits?

It is computationally heavy to calculate score densities using the model we have just described. Therefore, it is sensible to look first if a Central Limit Theorem (Laha and Rohatgi, 1979) applies to S_m (Equation 4.19) in the limit of a large number of dimensions m, and that the score distribution becomes Gaussian in this limit. If the answer to the question of *whether* a Gaussian limit appears is yes, then the next question is *when* it appears, i.e., for which values of m. Which values of m can be considered large?

In Appendix B.2 we prove that a Gaussian limit appears for the distribution of relevant document scores. Furthermore, we show that the distribution approaches the Gaussian quickly, such that corrections go to zero as $1/m$. Empirically, Gaussian shapes form at around $m = 250$. For the distribution of non-relevant document scores, we show in Appendix B.1 that a Gaussian limit is not likely, and if it appears, it only does so at a very slow rate with m. Empirically, we have never seen Gaussian shapes even for all dimensions resulting from massive expansion of queries.

4.4.2 Evaluation

Figure 4.1 shows empirical score data of non-relevant documents, Baumgarten's gamma distribution fit, and the density calculated by our model. The Rocchio-expanded query has around 400 dimensions. Training documents were *Ltu* weighted, while test documents were *Lu* weighted (Singhal, 1997). We approximated *Lu* and the dependencies introduced due to the large number of dimensions by

$$F_i(x) = F(x) = \frac{\log(x) - \log(a)}{\log(b) - \log(a)} , \quad 0 < a < b , \quad \forall i . \tag{4.24}$$

This means that the density function coming with F behaves as $1/x$ between a and b. We used the values that give a good fit with the empirical data: $a = 0.1$ and $b = 3.5$. We want to stress that, according to our observations, these parameters can be taken as constant for different queries of approximately the same length.

Our S-D curve is calculated with a *Monte Carlo* method (van Hameren, 2001), which is why it is plotted with steps. We have generated $1,000,000$ random $\boldsymbol{\omega}$ distributed following P_m, and made a histogram of their scores. The Monte Carlo algorithm for generating sets of m random numbers ω_i, for the F_i given by Equation 4.24, goes as follows:

- Generate a number x uniformly distributed in $[0, 1]$.
-
  - if $x > \varepsilon_i$, set $\omega_i = 0$.
  - if $x \leq \varepsilon_i$, generate a number y uniformly distributed in $[0, 1]$ and set $\omega_i = a\exp(\log(b/a)y)$.

The correctness of this algorithm can be shown using the *unitary algorithm formalism* explained in (van Hameren, 2001, pages 117–119). Obviously, this is not the most efficient way to obtain the curve of the score density P_{nr}. It is merely an easy way to obtain the curve without having to work out the equations involved.

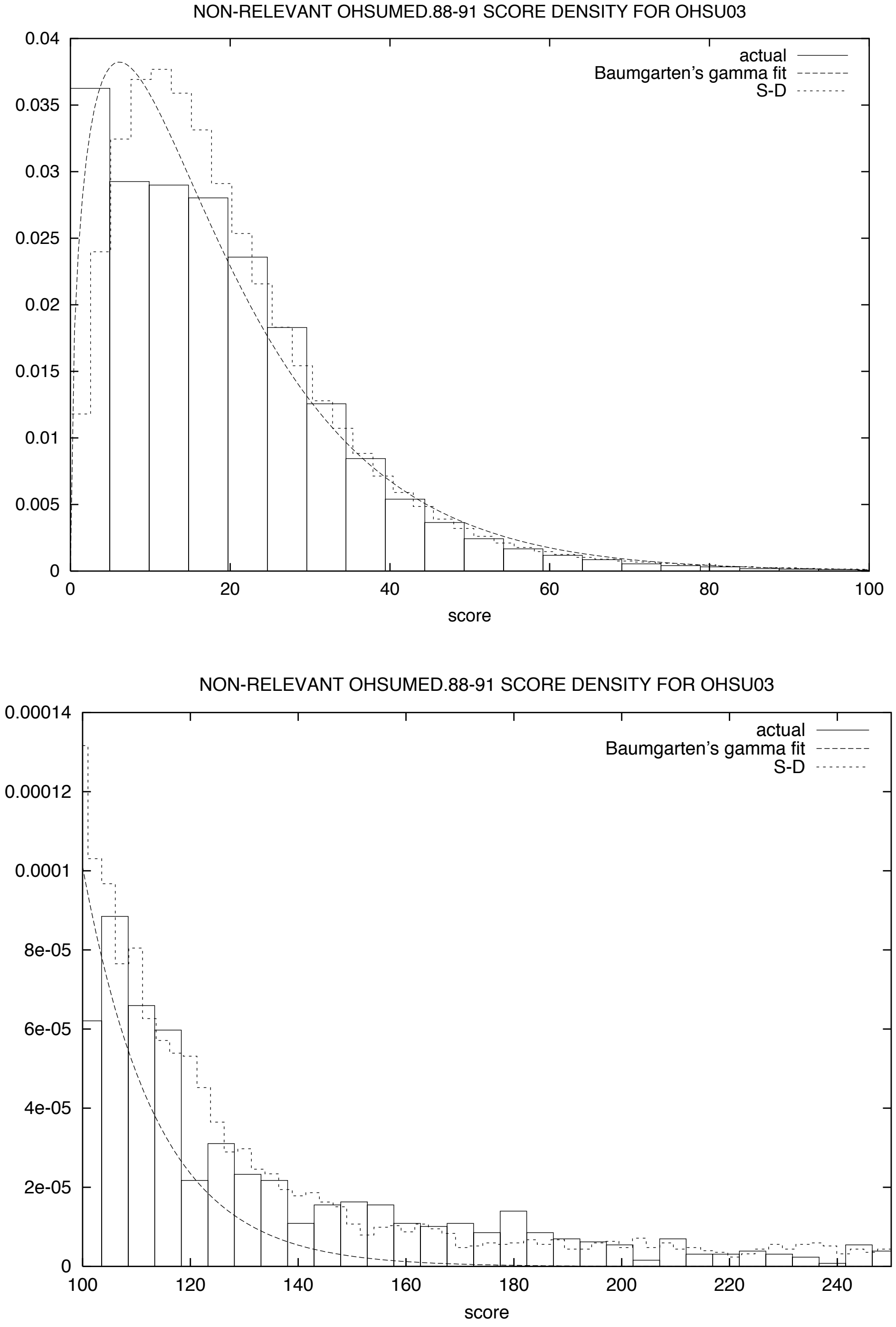


Figure 4.1: Score density of non-relevant documents (zero scores are excluded).

Equation 4.24 certainly does not correspond to *Lu* weights. It is just an *ad hoc* way of demonstrating how robust our model is: we have effectively obtained a very good fit on the empirical data, using the same F for all terms, and the effect of dependencies has turned out to be directly related to the *number* of dimensions, no matter which ones. The gamma distribution, nevertheless, gives a surprisingly good fit over a range of queries and dimensionalities. But our model is more accurate exactly where this is needed: *on the tail.*

4.5 Practical S-D Optimization

So far we have worked out an accurate optimization, disregarding how computationally expensive this may be. The goal of an optimization, however, is to improve filtering, and too much of a threshold accuracy may not capitalize in effectiveness (this still remains to be seen). It may be useful to see how the optimization can be applied more efficiently without sacrificing too much accuracy.

4.5.1 The Curse of Dimensionality

The optimization requires high dimensionality in queries to ensure a Gaussian central limit for P_{r}. Obviously, long queries can only be obtained from massive expansion through e.g. relevance feedback. One could argue against high dimensionality for efficiency reasons or due to the increased term dependencies introduced. Massive query expansion, however, has been shown to be effective (Buckley et al., 1994). Moreover, long queries are necessary when tracking relevance drifts, which are likely to occur in the retrieval environments we consider (Arampatzis and van der Weide, 2001). Above all, setting the thresholds right has proved to be critical for effectiveness in classification environments.

We will not recommend giving up on high dimensionality, since shorter queries may give zero scores for relevant documents truncating P_{r} at zero. Not only it is unclear how to estimate the parameters of a truncated distribution, but also our empirical data seem too irregular to be modeled by any known distribution. A Gaussian limit for P_{r} is convenient, and as we will see it simplifies calculations.

4.5.2 Approximations

P_{nr} has been defined numerically through a Fourier transform. A great simplification would be to fit a simple exponential of the form $c_1 \exp(-c_2 x)$ on the empirical score distribution, where c_1 and c_2 are the parameters determined by the fit. This approach has worked out well in (Arampatzis et al., 2000a), using a buffer of the top-50 scoring non-relevant documents and 5 bins. Figure 4.2 shows the empirical score distributions for TREC topic 352 on the Financial Times collection. The bar-charts represent the empirical score distributions of the relevant and non-relevant training documents. We collected these data as follows. First, we trained a classifier using all relevant documents and an equal number of the top-scoring non-relevant documents using the *query zone*[3].

[3] For the query zoning method, see Section 5.5.6.2 or (Singhal et al., 1997).

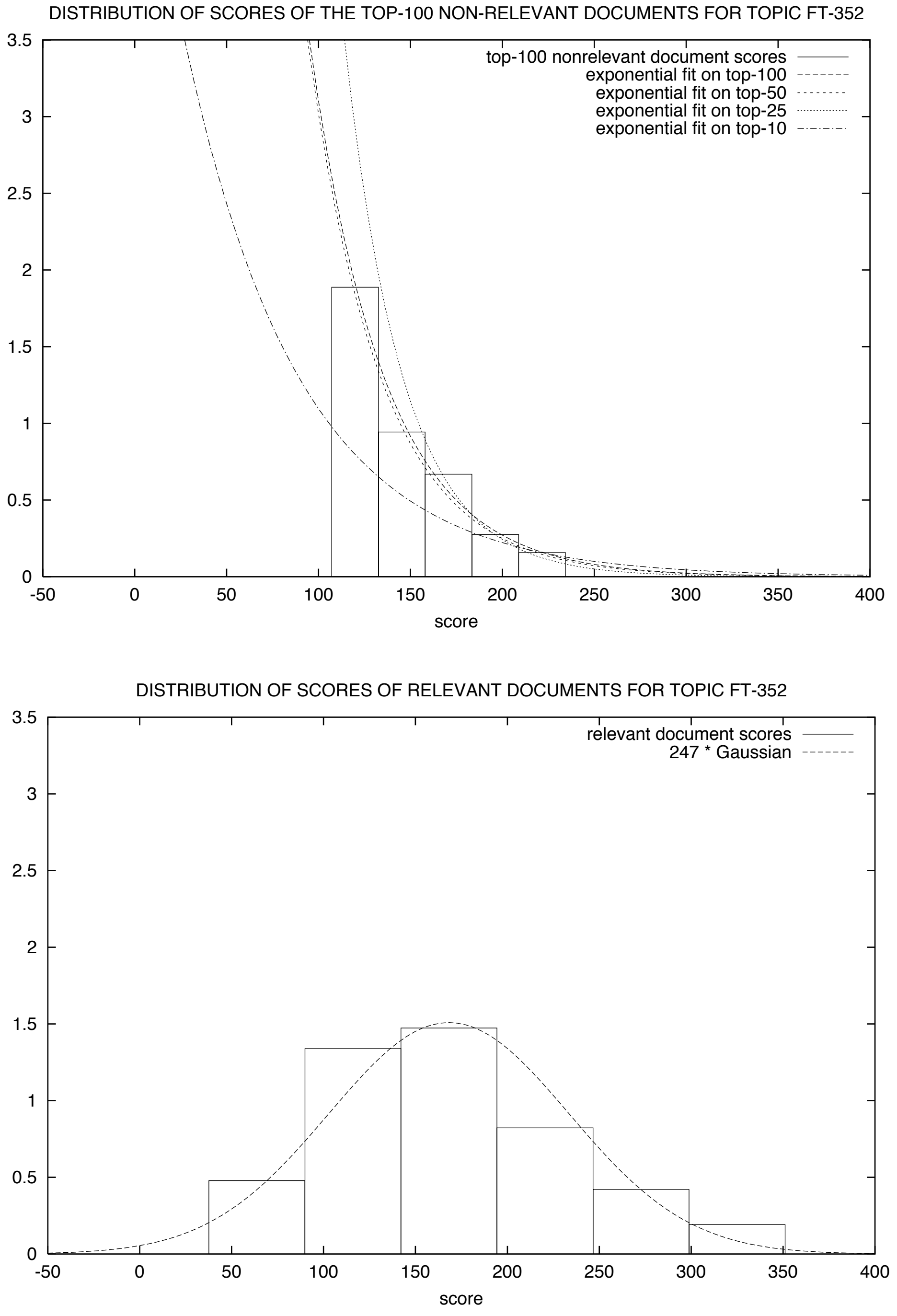


Figure 4.2: Empirical score distributions and the corresponding exponential (top) and Gaussian (bottom).

Then, we calculated the scores of relevant and non-relevant documents for the classifier. The lower plot shows the empirical distribution of relevant document scores, and the corresponding Gaussian multiplied by the number of scores. The upper plot shows the empirical distribution of the top-100 non-relevant scores, and exponential curves fitted on the top 100, 50, 25, and 10 scores. It seems that at least 50 or more scores are needed for an accurate threshold estimate. Similar curves for TREC topic 391 can be found in Appendix B.3.

An extra bonus of using an exponential is that, for linear measures, Equation 4.11 can be solved analytically, if P_{r} and P_{nr} are replaced by the corresponding Gaussian and exponential. Figure 4.3 shows the optimal T9U (i.e. a linear utility with $\lambda = 2$ and a fixed lower bound at -100; see Section 5.2.4) threshold, which is simply the score at which the densities P_{r} and P_{nr}, weighed as λr and $n - r$ respectively (Equation 4.11), intersect each other; we will give the complete solution in Section 4.5.3. For non-linear measures, however, Equation 4.9 has to be solved numerically and involves computing *error functions*. More about numerical methods can be found in (Press et al., 1992).

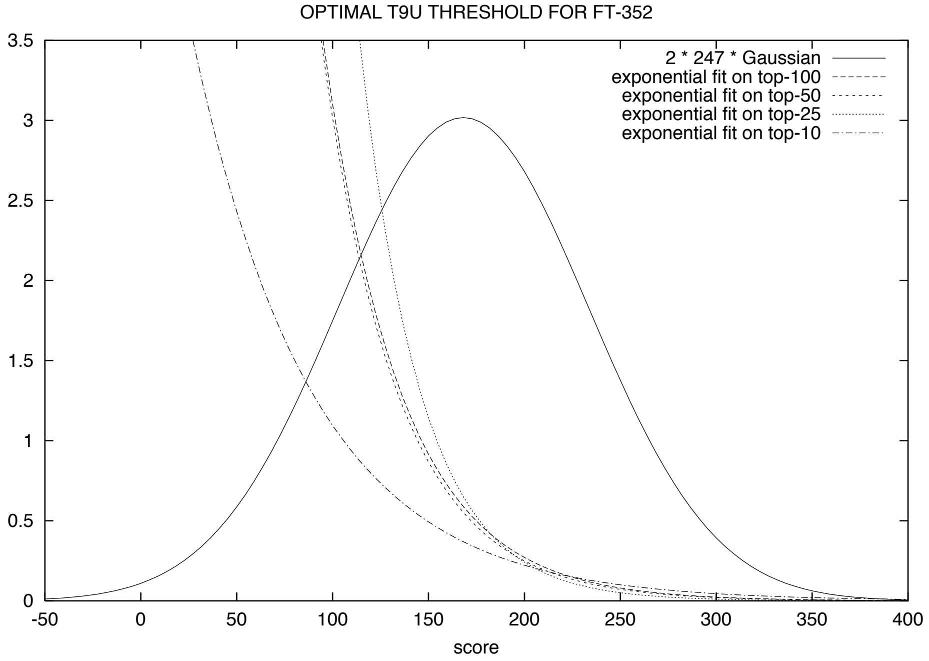


Figure 4.3: The optimal T9U threshold.

4.5.3 Optimizing Linear Utility Functions

In this section, we work out the practical optimization for the family of linear utility functions of Equation 4.1. The linearity of such measures allows for analytical solutions of Equation 4.9, since the integrals describing the document counts cancel out with the derivative of the measure.

As we have shown earlier in Section 4.3, the derivative of the family of linear utility functions is

$$\lambda \rho P_{\mathrm{r}}(\theta) = P_{\mathrm{nr}}(\theta), \quad \lambda = \frac{\lambda_3 - \lambda_1}{\lambda_2 - \lambda_4} \ , \quad \rho = \frac{r}{n-r} \ . \tag{4.25}$$

Let us now plug a Gaussian density for the relevant document scores

$$P_{\mathrm{r}}(\theta) = \frac{1}{\sqrt{2\pi\sigma_{\mathrm{r}}^2}} \exp\left(\frac{-(\theta - \mu_{\mathrm{r}})^2}{2\sigma_{\mathrm{r}}^2} \right) \ , \tag{4.26}$$

where μ_{r} is the mean score, and σ_{r} is the standard deviation. For the density of non-relevant scores, we will use an exponential of the form

$$P_{\mathrm{nr}}(\theta) = c_1 \exp\left(-c_2\theta\right) \ , \tag{4.27}$$

where the parameters c_1 and c_2 are selected so that P_{nr} is fitted on the right tail of the empirical distribution. Using Equations 4.26 and 4.27 on Equation 4.25, it results in

$$\lambda\rho \frac{1}{\sqrt{2\pi\sigma_{\mathrm{r}}^2}} \exp\left(\frac{-(\theta - \mu_{\mathrm{r}})^2}{2\sigma_{\mathrm{r}}^2} \right) = c_1 e^{-c_2\theta} \ .$$

By taking the logarithms of both sides and moving everything to the left, we get

$$\ln\left(\lambda\rho \frac{1}{c_1\sqrt{2\pi\sigma_{\mathrm{r}}^2}} \right) - \frac{1}{2\sigma_{\mathrm{r}}^2}(\theta - \mu_{\mathrm{r}})^2 + c_2\theta = 0 \ ,$$

which after a few routine calculations leads to the 2nd degree polynym

$$\begin{aligned} \frac{1}{2}a\theta^2 - b\theta + \frac{1}{2}c &= 0 \ , \\ a &= \frac{1}{\sigma_{\mathrm{r}}^2} \ , \\ b &= \frac{\mu_{\mathrm{r}}}{\sigma_{\mathrm{r}}^2} + c_2 \ , \\ c &= \frac{\mu_{\mathrm{r}}^2}{\sigma_{\mathrm{r}}^2} - 2\ln\left(\lambda\rho \frac{1}{c_1\sqrt{2\pi\sigma_{\mathrm{r}}^2}} \right) \ . \end{aligned} \tag{4.28}$$

The discriminant is $\Delta = b^2 - ac$, and the optimal threshold is

$$\theta = \begin{cases} (b - \sqrt{\Delta})/a \ , & \text{if } \Delta \geq 0 \ . \\ +\infty \ , & \text{if } \Delta < 0 \ . \end{cases} \tag{4.29}$$

Note that since the exponential corresponds to the top non-relevant scores, it does not extend accurately to low scores. Consequently, the optimization is more accurate when there is no contribution of N_- into the utility score, i.e. for $\lambda_4 = 0$.

4.5.4 Threshold Initialization

Our method relies on relevance information to estimate the corresponding curves, however, in some tasks (e.g., filtering) no such information may be available at the time of initiation. How should a threshold be initialized when there is sparse or no relevance information?

Let us assume that a stream of documents has already run for some time, when a new filtering $\boldsymbol{q}$ is issued. In principle, P_{nr} can be constructed with no relevance information, using the Fourier transform method and TPs calculated on *all* documents seen so far, since it is very close to the score density of all documents. The query itself can give an estimate of where P_{r} lies, e.g., $||\boldsymbol{q}||^2$ can be seen as the maximum relevant score. Some reasonable assumption for the standard deviation σ_{r} of P_{r} can produce a usable curve e.g. through an equation $\mu_{\mathrm{r}} = ||\boldsymbol{q}||^2 - 3\sigma_{\mathrm{r}}$, where μ_{r} is the mean of P_{r}.

4.5.5 Adaptivity and Soft Thresholds

A special problem that shows up in adaptive environments is that relevance information is becoming available only for documents retrieved. This may invalidate the score statistics required, and lead a system to a *selectivity trap* (Section 2.8). For instance, estimating a Gaussian from data which do not include its left tail (these are the data below the threshold), may over-estimate the threshold, retrieving no more documents.

A solution would be to use a *soft probabilistic threshold*, i.e., a document that scores at s, $s < \theta$, may still be retrieved by sampling it with a probability $P(\mathrm{rel}|s)$ given by Equation 4.13. Of course, the statistic that a document retrieved like this provides, should be weighted as $1/P(\mathrm{rel}|s)$. In this way, score statistics can be maintained more accurately, and selectivity traps can be avoided. The idea remains to be tested.

4.5.6 Incrementality

In general, means and deviations can be updated incrementally. In our context, however, every query update causes the scores of previously seen documents to change, suggesting that all scores should be re-calculated. Assuming a *static* W, in the sense that document weights do not depend on any statistics external to documents (e.g., documents are only *tf*-weighted), we show in Appendix B.4 that

$$\mu_{\mathrm{r}} = \frac{1}{r}\sum_{i=1}^{r}\langle \boldsymbol{q}, \boldsymbol{\omega}_i\rangle = \frac{1}{r}\langle \boldsymbol{q}, \sum_{i=1}^{r}\boldsymbol{\omega}_i\rangle \ . \tag{4.30}$$

Obviously, the sum of relevant document tuples is sufficient and can be updated incrementally. This way of calculating the mean score has been seen before in (Callan, 1998).

The variance σ_{r}^2 can be calculated via $\sigma_{\mathrm{r}}^2 = \mu_{\mathrm{r}}^{(2)} - \mu_{\mathrm{r}}^2$, where the mean of the squared scores is given by

$$\mu_{\mathrm{r}}^{(2)} = \frac{1}{r}\sum_{i=1}^{r}\langle \boldsymbol{q}, \boldsymbol{\omega}_i\rangle^2 = \frac{1}{r}\sum_{jk} q_j \left(\sum_{i=1}^{r}\omega_{ij}\omega_{ik}\right) q_k \ , \tag{4.31}$$

where e.g. ω_{ij} is the value of the jth component of document i. The proof of Equation 4.31 is given in Appendix B.5. The sum in the parenthesis can be represented by a 2-dimensional matrix $\mathbf{o}$ with components

$$o_{jk}^{(r)} = \sum_{i=1}^{r} \omega_{ij}\omega_{ik} \ , \tag{4.32}$$

which can be updated incrementally as

$$o_{jk}^{(r+1)} = o_{jk}^{(r)} + \omega_{(r+1)j}\omega_{(r+1)k} \ , \tag{4.33}$$

upon the arrival of document $\boldsymbol{\omega}_{r+1}$.

In the case where an exponential fit is used for P_{nr}, a small document buffer to hold the top-scoring retrieved non-relevant documents is indispensable because all scores should be re-calculated. If this buffer is full when a new non-relevant document is retrieved, the approach of ranking the buffered documents and discarding the lowest-scoring one has worked out well in (Arampatzis et al., 2000a), as we will see in Chapter 5. The Fourier transform method does not require a buffer.

4.6 Summary

We have developed a novel method for optimizing thresholds, namely, the *score distributional (S-D) threshold optimization*. The method is capable of optimizing any effectiveness measure defined in terms of the contingency Table 4.1. The analysis we have provided, we believe, is general enough to apply to a range of retrieval models, from probabilistic to vector space. Moreover, the method can be applied incrementally, a highly desirable feature for adaptive environments.

An earlier version of the S-D optimization has been tested in the context of the TREC-9 filtering task, and found to be very effective; we will report the empirical evaluations in Chapter 5. In this Chapter, we have revised the method so as to achieve better accuracy, especially in adaptive environments, and better incrementality. We have provided a range of choices, from very accurate and computationally expensive to practical and less expensive approximations. Whether the more accurate choices capitalize in improvements in classification effectiveness still remains to be seen.

Our work in modelling score distributions can be useful beyond threshold optimization problems. It can be applied to any retrieval environment that may use such distributions, e.g., distributed retrieval (Baumgarten, 1999), or topic detection and tracking (Spitters and Kraaij, 2000).

Chapter 5

The TREC Filtering Track

This chapter is based on our previously published work in (Arampatzis et al., 2000a). It describes our participation in the TREC-9 Filtering Track. For completeness, we give a brief introduction to TREC in Section 5.1, and focus on the Filtering Track in Section 5.2, especially on the setup of TREC-9. The reader familiar with TREC and the Filtering Track may proceed to Sections 5.3+ where our experience is described.

5.1 What is TREC

TREC[1] stands for Text REtrieval Conference, held annually since 1992 at the National Institute of Standards and Technology (NIST). Its purpose is to support Information Retrieval research by providing the infrastructure necessary for large-scale evaluation of retrieval methodologies. Although, at its beginning, the initiative focussed on text retrieval, it has expanded to include retrieval of other media, such as audio or video.

The workshop consists of a set of tasks known as *tracks*. Each track focuses on a particular variant of the retrieval task. The set of tracks (and their definitions) have varied over the years, depending on data availability and interest shown by participants. For example, the call for participation of TREC-10, which will be held in November 2001, specifies the following tasks:

Cross-language Retrieval. The track investigates the ability of systems to find documents that pertain to a topic regardless of the language in which the document is written.

Filtering. Documents are assumed to arrive one at a time. For each document, systems must make a binary decision for each topic whether the document should be retrieved for the topic or not.

Interactive. The track studies the interaction between users and retrieval systems.

Question Answering. As opposed to document retrieval, systems must return a text snippet containing the answer for a specific question.

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

Video Retrieval. The track investigates the content-based retrieval of digital video.

Web Retrieval. The track investigates the retrieval of web-pages.

Other tracks in previous years included Ad Hoc Retrieval, Spoken Document Retrieval, High Precision, Very Large Corpora, Natural Language Processing, and retrieval of documents in languages other than English, such as Chinese and Spanish. Organizations may choose every year to participate in any number of the specified tracks.

For each TREC, NIST provides test sets of documents and questions. Participants run their own retrieval systems on the data, and return their results to NIST. NIST judges the retrieved documents for correctness, and evaluates the results. The TREC cycle ends with a workshop that is a forum for participants to share their experiences. More about the TREC conferences can be found in the overview reports of the TREC Proceedings, e.g., (Voorhees and Harman, 1999), (Voorhees and Harman, 1998).

5.2 The Filtering Track

We will describe in this section, how the filtering track has developed over the years, and specify in more detail the setup used for TREC-9. We will give the definitions of the tasks, the datasets used for experiments, and evaluation measures. For the complete overview of the track see (Lewis, 1995b), (Lewis, 1996), (Hull, 1997), (Hull, 1998), (Hull and Robertson, 1999), and (Robertson and Hull, 2000).

5.2.1 Definitions of the Tasks

Consider a stream of documents already running for some time (the system has some history) when a user issues a query.

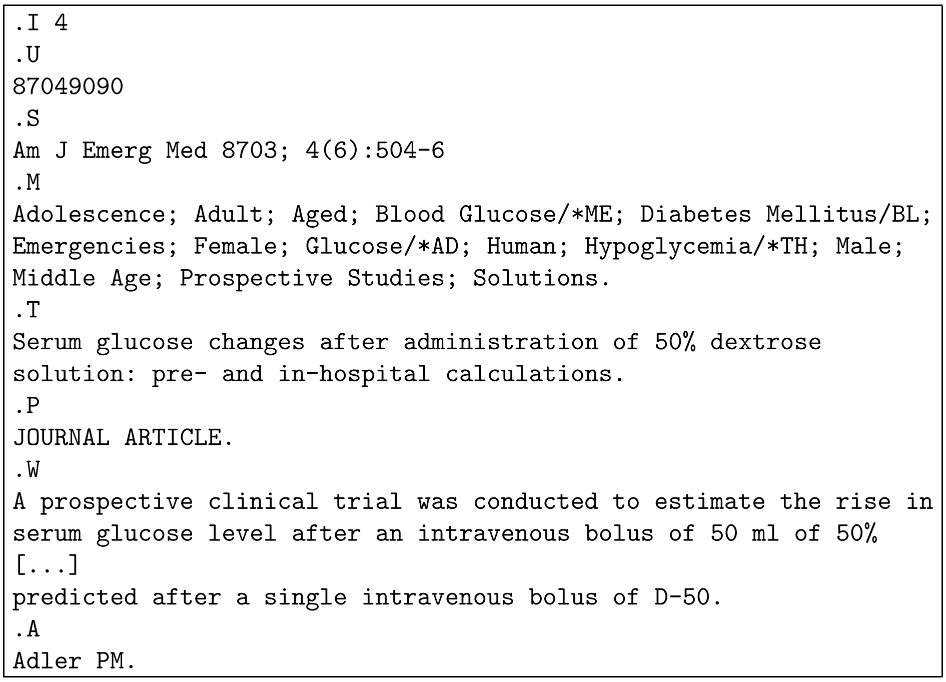
Adaptive Filtering. For the user query, a limited (or zero) number of documents from the history is given as being relevant, although many more may have been. A system should use only this relevance information, and try to find new relevant documents in the future. A binary decision whether to retrieve a document or not should be made as the document arrives, meaning that decisions cannot be postponed or retrospectively altered. For every document that the system retrieves, the user is assumed to give immediately a binary judgement of relevance for that document. This information may be used to update the filtering model. No user relevance judgements are provided for non-retrieved documents, since the user is assumed to have absolutely no idea of what the system rejects. Of course, the system may accumulate any other kinds of statistical data on all documents occurred in the stream.

Batch Filtering. The system starts with almost *complete* relevance judgements on the past, meaning that when the query is submitted almost *all* documents relevant to the query in the history of the system are known. In this respect, the system is given much more relevance information to begin with. There are two options on how to proceed. The first is not to adapt the filtering model at all in the future, while the second option is to proceed as in the aforementioned adaptive case. Systems that choose the second option are named *batch-adaptive*.

Routing. The system starts with the same relevance information as in batch filtering, but it is not allowed to adapt in the future. Furthermore, instead of making a binary decision, the system assigns retrieval scores to the incoming documents. The final output is a list of the top-1000 ranked documents.

In fact, systems have to process more than one user requests, which for practical reasons are all assumed as being submitted at the same time and last until the end of the incoming document stream.

Filtering was introduced as a separate track in TREC-4 to address a more difficult version of the routing track. The routing track has been performed since TREC-1. “Routing” in TREC has been defined rather unrealistically. Real-world routing applications require a system to a make binary decision of whether or not to retrieve an incoming document, not simply form a ranked list of documents. In this respect, TREC-4 introduced the filtering track, incorporating routing and extending it to batch filtering where binary decisions are required. The adaptive task was introduced in TREC-6 and it is admittedly of increased difficulty and realism. Since then, the task definitions above have been more or less the same.



```
.I 4
.U
87049090
.S
Am J Emerg Med 8703; 4(6):504-6
.M
Adolescence; Adult; Aged; Blood Glucose/*ME; Diabetes Mellitus/BL;
Emergencies; Female; Glucose/*AD; Human; Hypoglycemia/*TH; Male;
Middle Age; Prospective Studies; Solutions.
.T
Serum glucose changes after administration of 50% dextrose
solution: pre- and in-hospital calculations.
.P
JOURNAL ARTICLE.
.W
A prospective clinical trial was conducted to estimate the rise in
serum glucose level after an intravenous bolus of 50 ml of 50%
[...]
predicted after a single intravenous bolus of D-50.
.A
Adler PM.
```

Figure 5.1: A document from OHSUMED.

5.2.2 Document Streams — OHSUMED

For the TREC-9 filtering track, the OHSUMED collection was used as a test dataset. OHSUMED has been used before for IR experiments, see e.g. (Hersh and Hickam, 1994), (Hersh et al., 1994), or (Lewis et al., 1996). The collection is a set of 348,566 references from MEDLINE[2], the on-line medical information database, consisting of titles and/or abstracts from 270 medical journals over a five-year period (1987-1991). The available fields in documents are: title, abstract, MeSH indexing terms, author, source, and publication type. Some abstracts are truncated at 250 words and some references have no abstracts at all (titles only). Figure 5.1 shows an example document.

Other document collections that have been used before in TREC as experimental document streams are: the Foreign Broadcast Information Service (FBIS) corpus in TREC-6, the Associated Press (AP) newswire collection in TREC-7, and the Financial Times (FT) in TREC-8.

5.2.3 Topics and Relevance Judgements

Two primary sources of filtering topics were used for the TREC-9 filtering track:

1. A subset of the original query set developed by (Hersh and Hickam, 1994) for their experiments.
2. A set of MeSH terms and their definitions[3].

The existing OHSUMED topics describe actual information requests, but the relevance judgements probably do not have the same coverage provided by the TREC *pooling* process[4]. This simply means that there may be more relevant documents than the existing relevance judgements suggest. The MeSH terms do not directly represent information requests, they are rather controlled indexing terms, and the assessment is more or less complete. Our group has experimented only with the OHSUMED topics, so next we will focus on these.

The topic statements are provided in the standard TREC format and consist of `<title>` and `<desc>` (= description) fields only. The meaning of these fields for the OHSUMED topics is the following:

- `<title>` = patient description.
- `<desc>` = information request.

Figure 5.2 shows an example OHSUMED topic.

The test collection was built as part of a study assessing the use of MEDLINE by physicians in a clinical setting (Hersh and Hickam, 1994). Novice physicians using MEDLINE generated 106 queries. Only a subset of 63 of those queries were used in the TREC-9 filtering track. Before the physicians searched, they were asked to provide a statement of information about their patient as well as their information need.

[2] http://www.medline.com

[3] http://www.nlm.nih.gov/mesh/

[4] For TREC's pooling process, see e.g. (Voorhees and Harman, 1999).

```
<top>
<num> Number: OHSU27
<title> 75 year old with diabetes and hypertension with Q waves on EKG
<desc> Description:
differential diagnosis of U waves
</top>
```

Figure 5.2: An example OHSUMED topic.

Each query was replicated by four searchers, two physicians experienced in searching and two medical librarians. The results were assessed for relevance by a different group of physicians, using a three point scale: definitely, possibly, or not relevant. Over 10% of the query–document pairs were judged in duplicate to assess inter-observer reliability. For evaluation in TREC-9, all documents judged as either possibly or definitely relevant were considered relevant. However, systems had the option to distinguish between these two categories during the learning process.

The OHSUMED 1987 documents were intended for training purposes only. For batch filtering and routing, all evaluated documents from the 1987 collection were given as known. For adaptive filtering, only two documents judged as definitely relevant were given for each topic. The training samples extracted for adaptive filtering were selected by random sampling. All runs were allowed to use the OHSUMED 1987 collection for generating collection summary statistics (such as IDF) or other purposes.

5.2.4 Evaluation Measures

TREC's main concern has been effectiveness rather than efficiency. For each subtask, systems have to return a set of documents per topic for evaluation. For the filtering tasks, the retrieved sets are assumed to be unordered and of arbitrary size. The retrieved sets of the routing task are limited to the top-1000 documents per topic.

The routing task has been traditionally evaluated according to the *average uninterpolated precision*[5], i.e. the sum of precision values at every position of the rank divided by 1000 and averaged over all topics. Average precision is a single-valued measure which combines both the precision and recall of a system; it amounts to the area below the recall–precision curve. In early TRECs, until TREC-5 and 6, more detailed evaluations were being reported in the form of 11-point interpolated recall–precision and recall–fallout, but as the focus of the track has moved from routing to filtering these measures have been abandoned.

Since the filtering tasks return unordered sets of documents, not rankings, different effectiveness measures have been used. These have mainly been utility functions; the quality of filtering is computed as a function of the benefit of retrieving a relevant document and the cost of retrieving a non-relevant one. In their simplest form, utility functions have been linear, however, non-linear measures were tried as well in TREC-8. The non-linear utilities assumed *diminishing returns* from relevant documents, for exam-

[5] The term *average uninterpolated precision* has been usually (and misleadingly) used to refer to the average uninterpolated precision *averaged* over all topics.

ple, the 100th relevant document retrieved provides less benefit than the 10th. Only a few participants optimized their runs for those measures; many participants felt that the non-linear measures did not model a user's behaviour very well. Another experimental measure tried in TREC-6 was the *average set precision (ASP)*, i.e. the product of precision and recall for the retrieved set. Table 5.1 summarizes the evaluation measure trends in the TREC filtering track. Developing appropriate effectiveness measures for filtering continues to be an important part of the track.

TREC	2	3	4	5	6	7	8	9	
Routing	Average Uninterpolated Precision								
	11-point Recall–Precision								
	11-point Recall–logFallout								
Filtering			Linear Utilities						
					ASP		non-L. Utilities	T9P	

Table 5.1: Effectiveness measure trends in TREC.

For TREC-9, the routing task was evaluated with average precision. The filtering tasks were evaluated according to a linear utility measure and a precision-oriented measure; we will describe these below in more detail. There had been no additional relevance judgements on the retrieved documents[6]; all evaluation was done on the basis of the existing relevance judgements. All runs were evaluated based on the full document test set, OHSUMED 1988-91.

5.2.4.1 TREC-9 Utility — T9U

For TREC-9, a single specific linear utility function was used, with a fixed lower bound MinU. The lower bound MinU ensures that an individual topic which performs really badly will not dominate the average. It must depend on the time period. The specific figure chosen for the OHSUMED topics is −100, and is adjusted *pro rata* for evaluating adaptive filtering over shorter periods. This is equivalent to selecting approximately 2 non-relevant documents per month and no relevant documents. Using the variables of the contingency Table 4.1, the measure is defined as follows:

$$\begin{aligned} \mathrm{T9U} &= \begin{cases} 2R_+ - N_+ \ , & \text{if } (2R_+ - N_+) > \mathrm{MinU} \ . \\ \mathrm{MinU} \ , & \text{otherwise} \ . \end{cases} \\ \mathrm{MinU} &= -100 \ , \quad \text{for 4 years or pro rata adjusted} \ . \end{aligned} \tag{5.1}$$

Raw T9U figures were averaged across topics, with no normalization.

5.2.4.2 TREC-9 Precision-oriented Measure — T9P

The aim of this measure is to stress precision, while demanding a minimum number of documents, MinD, to be selected for each topic. As with the lower bound on utility, MinD must depend on the time period. It was set at 50 documents over the 4-year test period, or approximately 1 per month. The measure is identical to precision except that

[6] Additional relevance assements are usually done in TRECs according to the pooling methodology.

the denominator is constrained by MinD, thus penalizing systems which retrieve less than MinD documents:

$$\begin{aligned} \text{T9P} &= \frac{R_+}{\max(\text{MinD}, (R_+ + N_+))} , \\ \text{MinD} &= 50 , \quad \text{for 4 years or pro rata adjusted.} \end{aligned} \tag{5.2}$$

The T9P measure was calculated for each topic and averaged across topics.

5.3 On the TREC-9 Filtering Track

As first-time TREC participants, we participated in all three subtasks — adaptive, batch, and routing — while concentrating mainly on adaptive tasks. We have made use of two different systems:

- FilterIt, for the adaptive and batch-adaptive[7] tasks: a pure adaptive filtering system developed in the context of our TREC-9 participation. It is based on the Rocchio algorithm.

- LCS (Ragas and Koster, 1998), for the routing and batch filtering tasks: a multi-classification system based on the Winnow algorithm.

Task	Topics	Optimized for	System	Run-tag
adaptive	OHSUMED	T9U	FilterIt	KUNa1T9U
adaptive	OHSUMED	T9U	FilterIt	KUNa2T9U
adaptive	OHSUMED	T9P	FilterIt	KUNa1T9P
adaptive	OHSUMED	T9P	FilterIt	KUNa2T9P
batch-adaptive	OHSUMED	T9U	FilterIt	KUNbaT9U
batch	OHSUMED	T9U	LCS	KUNb
routing	OHSUMED	—	LCS	KUNr1
routing	OHSUMED	—	LCS	KUNr2

Table 5.2: TREC-9 filtering runs submitted by KUN.

In adaptive filtering, our contribution has been threefold. Firstly, we have investigated the value of retrieved documents as training examples in relation to their time of retrieval. For this purpose we have introduced the notion of the half-life of a training document. Secondly, we have introduced a novel statistical threshold selection technique for optimizing linear utility functions. The method can be also applied for optimizing other effectiveness measures as well, however, the resulting equation may have to be solved numerically. Thirdly and most importantly for adaptive long-term tasks, we have developed a system that allows incremental adaptivity. We have tried to minimize the computational and memory requirements of our system without sacrificing its accuracy. In the batch and routing tasks, we have experimented with the use of the Winnow algorithm, including a couple of small improvements.

[7] We see the batch-adaptive task as an adaptive rather than a batch filtering task.

From the two topic-sets given, we have experimented only with the 63 OHSUMED queries. We did not submit any runs on the 4904 MeSH topics; these were simply too many to be processed by our present systems in a reasonable time and space. All experiments were done using a keyword-based representation of documents and queries, with traditional stemming and stoplisting, although our long-term intention is to use phrase representations (Chapter 6), and apply more sophisticated term selection methods (Chapter 3). Table 5.2 summarizes our official TREC-9 runs.

Next, we will briefly describe the pre-processing applied to the data. The FilterIt and lcs systems are described in Sections 5.5 and 5.6, respectively. In Section 5.7 we give an overall view to how our systems performed in relation to other participants.

5.4 Stream and Topic pre-processing

We used only the title and abstract fields (.T and .W fields in Figure 5.1) of the OHSUMED documents; their MeSH-headings were discarded. The pre-processing of the documents and topics was minimal and quite traditional. It consisted of the following steps:

1. Replacement of all non-letters by spaces.
2. Deletion of all one-letter words.
3. Lowercasing.
4. Stoplisting[8].
5. Stemming[9].
6. Deletion of all one-letter stems.
7. DF-stoplisting: removal of the top-100 stems with the highest document frequencies in OHSUMED 1987, only for the non-adaptive tasks. For the adaptive tasks, we did not remove any stems; incremental *idf* (see Section 5.5.3) does not combine well with DF-stoplisting.

The titles and descriptions of topics were processed in the same way.

In summary, our pre-processing was quick-and-dirty. There was no special treatment of proper names, all numbers were lost, and we made no use of multi-word terms such as phrases or word clusters. Moreover, we used no external resources such as online dictionaries or thesauri.

[8] We used the standard stoplist of the SMART system, `english.stop`, available from: `ftp://ftp.cs.cornell.edu/pub/smart/`

[9] We used the Porter stemmer of the `Lingua::Stem` (version 0.30) library extension to Perl.

5.5 The FilterIt System

The FilterIt system, which we used for performing the adaptive and batch-adaptive tasks, has been developed in the context of our TREC-9 participation. It is a pure adaptive filtering system based on Rocchio's method (Rocchio, 1971) (Section A.4). Rocchio's method performs well in a situation where only a few training documents are available, see e.g. (Ragas and Koster, 1998), and this is exactly the case in the adaptive task. In such a situation, the initial query becomes important and the method can moreover deal in a suitable way with the topic descriptions.

We have modified the formula traditionally used for relevance feedback in order to allow for weighing of training documents according to their time-stamps. Moreover, the implementation of the algorithm we will present, allows very accurate incremental training of classifiers, without using any document buffers, so its memory and computational power requirements are low. In order to further limit the memory requirements of our system per topic, we also use a form of on-the-fly term selection.

Our system adapts queries and thresholds independently for each topic, meaning that the filtering model for a topic is updated after the retrieval of every single document for that topic. In the runs optimized for the T9P measure, threshold adaptations are even triggered independently of document retrievals.

For optimizing the filtering thresholds, we have introduced a new statistical technique which takes into account the relative density of relevant to non-relevant documents seen in the stream, and their score distributions. Most of the quantities that our technique requires can be updated incrementally, but a small document buffer seems unavoidable.

In the rest of this section we will expand on all the above.

5.5.1 Incremental Query Training

The version of Rocchio's method traditionally used for relevance feedback is

$$Q = \alpha\, Q_0 + \beta \frac{1}{|\mathcal{R}|} \sum_{D \in \mathcal{R}} D - \gamma \frac{1}{|\mathcal{N}|} \sum_{D \in \mathcal{N}} D\ , \tag{5.3}$$

where Q_0 the initial query, $\mathcal{R}$ and $\mathcal{N}$ the sets of relevant and non-relevant documents respectively, and $|.|$ denotes the number of elements in a set. The parameters α, β, and γ control the relative contribution of the initial query, and that of the relevant and non-relevant documents to the new query Q. All components which end up with negative weights in Q are removed.

The initial query and the documents are usually represented by vectors weighted in a *tf.idf* fashion[10]. While the *tf* components are usually independent of corpus statistics, the *idf* components depend on the collection. Since the whole collection in filtering is not available in advance, the *idf* components should be updated over time (*incremental idf*).

[10] *tf.idf* denotes here the *family* of weighting schemes for which the value of a term increases with its frequency in a document or query and decreases with its frequency across the collection. In practice, *tf* and *idf* are implemented by some monotonically increasing (non-linear) functions of the corresponding frequencies. We will give our precise choice of these functions in Section 5.5.3.

Therefore, it would be more suitable for filtering to keep these quantities separately. As a result, queries and documents in our system are only *tf*-weighted, e.g., a document D_i is represented by

$$D_i = [\mathit{tf}_{i1}, \ldots, \mathit{tf}_{iK}] \ , \tag{5.4}$$

where K the total number of terms known by the system at one point in time. Any document or query is a sparse array since it contains far less non-zero components than K, so they are implemented by *hash arrays*.

Since all vectors are only *tf*-weighted, we have moved the impact of *idf*s into the similarity function, which for a query Q and a document D has been defined as:

$$S(Q, D) = Q \, \mathit{IDF} \, D^T \ , \tag{5.5}$$

where IDF is the diagonal matrix $\mathit{diag}(\mathit{idf}_1, \ldots, \mathit{idf}_K)$, and X^T denotes the transposed array of X. Such an implementation allows, at any time, the usage of the latest *idf* values.

Now, Equation 5.3 can be calculated incrementally by simply re-writing it as

$$Q_n = \alpha \, Q_0 + \beta \frac{1}{R_n} B_n - \gamma \frac{1}{N_n} C_n \ , \tag{5.6}$$

where B_n, C_n are the accumulated sums of the term frequency vectors of relevant and non-relevant documents respectively, and R_n, N_n are the numbers of documents in each category[11]. When document D_n is retrieved, Q_n is calculated in two steps. First, all time-dependent quantities (everything on the right side of the formula which has the subscript n) in the last formulation are updated. Then, the query Q_n is calculated using the updated quantities.

Summarizing, the architecture we have just described allows the most accurate incremental training with Rocchio. No training documents have to be discarded, as would have been necessary in a *sliding window* adaptive system. Moreover, no document buffers are necessary, except B_n and C_n in which all training documents are accumulated. In order to achieve all these, the only requirement is that *tf*s are *static* in the sense that they can be calculated only once when a document arrives. As we have seen in Section 4.5.6, this feature allows for incrementality in the calculation of the mean relevant document score and variance — parameters necessary for optimizing filtering thresholds.

Of course, there is another minor concession we make here, that is to allow counting registers of infinite width (the values of the components of B_n, C_n, and the variables R_n, N_n can grow up to infinity). Double precision arithmetic approximates this assumption well. In any case, when a number approaches the maximum width, all quantities can be divided by a constant without invalidating the model.

[11] The convention we use for the subscript n is: n is the total number of training documents available (relevant and non-relevant). Training documents are the ones given at the time of bootstrapping (as for the batch-adaptive task), and all retrieved ones during filtering since their relevance judgment can be seen. Thus, Q_n is the classifier built using n training documents. If r of them are relevant, then $R_n = r$ and $N_n = n - r$, and B_n, C_n contain the sum of r and $n - r$ document vectors respectively.

5.5.2 Convergence, Responsiveness, and Decay

The goal of the incremental training we have described so far is to gradually converge to a perfect classifier. All training documents, irrespective of their time of retrieval, are taken into account with equal importance in constructing the classifier. Systems that implement this kind of converging adaptivity we call *asymptotically adaptive*. The use of an asymptotically adaptive system for filtering implicitly assumes that topics are *stable*, i.e. there are no *topic drifts*.

If there are topic drifts, the position of the perfect classifier moves in the document space. Therefore, it is beneficial for a filtering system to be capable of tracking a topic rather than converging. This capability can be achieved by weighing training documents that are retrieved recently more heavily. We call such systems *locally adaptive*. The choice between local adaptivity and asymptotic adaptivity should be made depending on whether *convergence* or *responsiveness* is more important. More about various forms of adaptivity for filtering systems and the nature of topics in filtering have been discussed in Chapter 2.

In TREC-9, topics are assumed to be stable, suggesting that an asymptotic behaviour would be more proper. However, the OHSUMED collection consists of documents collected in a period of five years and it is likely that for a topic the content of its relevant documents changes over the years, e.g., think of new treatments developed for the same sickness. The effect of such *document content drifts* is equivalent to *user interest drifts* in the sense that the idea of *relevance* changes. Consequently, we experimented with a locally adaptive system.

In order to weigh training documents differently, we replace the average vectors in the Rocchio formula of Equation 5.3 with *weighted averages*. This does not invalidate the motivation of the formula. For instance, the average vector of relevant documents becomes

$$\frac{1}{|\mathcal{R}|}\sum_{D\in\mathcal{R}} D = \frac{1}{\sum_{i:D_i\in\mathcal{R}} l_i}\sum_{i:D_i\in\mathcal{R}} l_i D_i \ , \tag{5.7}$$

where l_i represents the weight with which the document D_i contributes to the average.

In Section 2.5, we have seen that a heavier weighting of recently retrieved training documents can be implemented by a *decay* operation with *half life h*. Whether the initial query Q_0 should decay or not depends on the nature of a topic. For a drifting user interest, Q_0 should decay. For a stable interest with document content drifts (as we have argued to be true for TREC-9), any of the two choices can be motivated (it rather depends on how Q_0 is formulated). For our official TREC-9 runs, we chose to decay or gradually eliminate Q_0. We will come back to the subject of initial query elimination in Section 5.5.6.1.

The decay operation can be performed incrementally. When D_n is retrieved, and assuming that it is found to be relevant, then it is easy to show that average vectors, e.g., the one of relevant documents, can be updated as:

$$\frac{1}{R_n}B_n = \frac{1}{lR_{n-1}+1}\left(lB_{n-1}+D_n\right) \ , \quad l = 0.5^{(g/h)} \ ,$$

where $g = t_n - t_{n-1}$ stands for the elapsed time since the previous query update (i.e., since

Q_{n-1} was calculated), and h is the half life. Therefore, when a document is retrieved, all time-dependent quantities of Equation 5.6 are multiplied by the current decay factor l before they are updated with the new document. To maintain correct decaying weights, even the quantities which are not going to be updated have to be multiplied, e.g. even if D_n is relevant, $N_n = lN_{n-1}$ and $C_n = lC_{n-1}$.

In TREC-9, time was estimated on the number of documents seen in the stream. It was given that the stream produces, on average, around 6,000 documents per month. Therefore, for a half life of m months, we set $h = 6,000m$, and g is simply the number of documents filtered since the previous query update.

5.5.3 Term Weighting — *Ltu*

For term weighting, we "borrow" from (Singhal, 1997) the *Ltu* formula:

$$\begin{aligned} L &= \frac{1+\log(f)}{1+\log(\text{average } f \text{ in document or query})} \;, \\ t &= \log\left(\frac{N+1}{df}\right) \;, \\ u &= \frac{1}{0.8+0.2\frac{\text{number of unique terms in document or query}}{\text{average number of unique terms per document}}} \;, \\ Ltu \text{ weighting} &= L \times t \times u \;, \end{aligned} \tag{5.8}$$

where f is the frequency of the term in the document or query, df is the number of documents in which the term occurs from a collection of N documents in total. In the *Ltu* weighting scheme, L is the term frequency factor, t is the inverted document frequency factor, and u is the length normalization of the document or query.

The N and df values were initialized from OHSUMED 1987. Then we used *incremental idf*: upon the arrival of a new document and before any other calculation is performed, the df values are updated and N is incremented by one[12]. In this way, for all document terms we have $df > 0$ and the t factor can be defined. For any query terms with $df = 0$, we set $t = 0$.

The application of the *Ltu* formula in adaptive filtering presents a small problem. The average number of unique terms per document changes over time, therefore, the term weights of past documents should be re-calculated as well. We chose to calculate this average document length on OHSUMED 1987 and assume that it will not change in the future. This allows the calculation of the u factor once and for all, when a document arrives. The assumption that the average document length will remain the same in the future is not far from reality for the OHSUMED collection, since there is no special reason why medical researchers should write abstracts of different lengths over time.

[12] Actually, the probability of a document containing the term may itself be time-varying. Consequently a temporally local estimation of t, e.g. using an exponential decay for N and df in the same way as for training documents, would be more proper. We have not used such a scheme for filtering the OHSUMED stream, because we believe that its composition (the kind of documents it contains) is not altered considerably during its time-span.

Summarizing and using our notation, the exact form of the term weighting we used is:

$$tf = L \times u' , \quad idf = t , \tag{5.9}$$

where u' is the same as u but with the average document length fixed on its OHSUMED 1987 value (that was 40.8 keywords after the pre-processing). This form presents *static tf* components and *dynamic idf*s. These features allow for incrementality in training and threshold optimization, as we have shown in Sections 5.5.1 and 4.5.6 respectively.

5.5.4 On-the-fly Term Selection

It is empirically known that as the size of a corpus grows, the number of unique words seen grows with the square root of the number of documents. In the case of multi-word terms (phrases), the number of such enriched terms grows even faster. Therefore, the number of components of B_n and C_n vectors grows, at least, with the square-root of the number of retrieved documents n. To limit the size of these vectors we use term selection.

In fact, term selection is more critical for the S-D threshold optimization we have introduced in Chapter 4. The incremental application of the optimization requires matrices as large as the *square* of the size of B_n or C_n (see Section 4.5.6), consequently the memory requirements may explode early on, if no term selection is used.

Term selection is applied for each topic independently, before every incremental update of the corresponding query. Our *on-the-fly*[13] term selection scheme consists of the following steps. First, a query is constructed using information only from relevant instances and the current *IDF* matrix:

$$Q_{n,\mathrm{rel}} = \left(\alpha\, Q_0 + \beta \frac{1}{R_n} B_n \right) \mathit{IDF} \, . \tag{5.10}$$

Then, we rank all terms of $Q_{n,\mathrm{rel}}$ according to their weight, and select only the top-k ones and the terms occurring in Q_0. The rest of the terms are discarded and removed from all quantities kept by the system for the topic (e.g., B_n and C_n). Then, Q_n is calculated using the reduced data.

This technique limits the memory required for filtering a topic. However, the size of the *IDF* matrix still grows by the time, as previously unseen terms occur in documents of the stream. We consider *IDF* as *stream data* rather than *topic data*, since it is the same for all topics being filtered at any point in time. Therefore, we do not limit its size.

5.5.5 Optimizing T9P

The S-D threshold optimization can be applied to optimize T9P. However, in this case Equation 4.9 does not have analytical solutions, therefore it has to be solved numerically. Regrettably, we did not do that.

[13] "On-the-fly" means that term selection is repeated during filtering (every time a new training document becomes available via feedback) in contrast to one-time term selection (just before training classifiers) in classification/categorization tasks.

The technique we used, lowers the threshold after every "quiet" month with respect to how many documents are missing according to the pro-rata adjusted MinD value. It goes as follows:

1. Right after a query update, start collecting the document scores in the range $[\mu_{\text{nr}}, \theta]$, where μ_{nr} is the mean score of the N_+ documents and θ the optimal S-D threshold for $U = R_+ - N_+$.

2. If after one month of documents nothing is retrieved, calculate how many should have been retrieved by the current time (pro-rata).

3. Check how many are missing:
   $m =$ pro-rata−retrieved.

4. If $m > 0$, lower the threshold to s_m, where s_m the top-m score seen below θ.

The method works, in the sense that it retrieves around MinD documents or more. Moreover, it retrieves the ones that score the highest. It assumes, however, that the distribution of relevant documents in the stream is uniform (or their relative density is approximately constant), in general a false assumption. Another drawback of the method is that it optimizes the threshold for $U = R_+ - N_+$ and not for precision. All of these, we believe, make our submitted T9P runs moderately satisfactory.

After all, we should have at least tried to solve Equation 4.9 numerically. Although analytical formulas are mathematically more elegant, in practice, numerical methods are efficient and easy to implement.

5.5.6 Experiments with FilterIt

The FilterIt system presents two features which we are interested in comparing their effectiveness with other systems: the practical S-D threshold optimization for linear utility functions (see Section 4.5.3), and the decay of training documents (see Section 5.5.2). The tuning parameters were numerous, and the runs allowed for submission to TREC-9 were limited to 4 for adaptive and to 2 for batch filtering (including batch-adaptive). Moreover, we submitted one of the two batch filtering runs with the lcs system described in Section 5.6. These limits do not allow extensive comparisons, and some choices had to be made.

Our strategy in deciding what to submit was as follows. For two of the four adaptive runs we did not use any of the two features but rather conventional techniques. In this way, we expected to have at least two runs with conventional effectiveness, in case our techniques would have failed. The other two adaptive and the single batch filtering runs combine all the new features. All parameters were set at "safe" values, as these were determined by our experiments with the Financial Times (FT) collection. More aggressive settings have yielded better effectiveness on FT, however, we do not believe that these generalize to all collections.

5.5.6.1 Rocchio Parameters and Initial Query Elimination

All adaptive runs use $\alpha = \beta = \gamma$ for Rocchio. These tasks start with a query and only 2 relevant training documents. In pilot runs on FT, traditional settings with $\alpha < \beta$ seemed to overfit the classifiers on those 2 relevant documents. Therefore, such small training sets should not be trusted and the initial query Q_0 should be weighted fairly high, e.g., as high as $\alpha = \beta$. As a filter is collecting more and more relevant documents, the contribution of the initial query can gradually be eliminated. Consequently, we moreover multiply Q_0 with $10/(R_n + 10)$ while calculating the new query Q_n. We do not use such an *initial query elimination* for the runs with decay since the initial query decays anyway.

For the batch-adaptive task, α is set at one-fourth of β. Since larger training sets are given for this task, the danger of overfitting is smaller. When using *query zones*[14], (Singhal et al., 1997) have shown that $\beta = \gamma$ is a reasonable setting. This explains why we set $\beta = \gamma$ also for the adaptive tasks. Thresholding document scores during filtering can been seen as a form of *on-the-fly query zoning*. Any non-relevant documents retrieved in this way are indeed the most interfering with the query. This setting has worked out well for us in our experiments on FT.

5.5.6.2 Submitted Runs

Table 5.3 summarizes the runs we submitted, their parameter settings, and the final results obtained.

Task	adaptive				batch-adapt.
Run	KUNa1T9U	KUNa2T9U	KUNa1T9P	KUNa2T9P	KUNbaT9U
Rocchio	$\alpha = \beta = \gamma$				$4\alpha = \beta = \gamma$
Q zone	no — *on-the-fly*				top-r
Q_0 elim.	$10/(10 + R_n)$	no	$10/(10 + R_n)$	no	no
θ for	T9U	T9U	T9P	T9P	T9U
method	$(\mu_{\mathrm{r}} + 2\mu_{\mathrm{nr}})/3$	S-D	$(\mu_{\mathrm{r}} + \mu_{\mathrm{nr}})/2$	S-D	S-D
half life	∞	2 yrs	∞	2 yrs	2 yrs
TS-cutoff	—	500	—	500	500
Result	+16.8	+17.3	0.258	0.231	+19.4

Table 5.3: Settings and results for the (batch-)adaptive submitted runs.

KUNa1T9U and KUNa1T9P do not use decay, term selection, or the threshold optimization described in this article. The threshold per topic is set at the midpoint of the average scores of relevant and non-relevant documents. In fact, for KUNa1T9U we set thresholds at one-third of the distance between the non-relevant and the relevant mean score to reflect the fact that the gain of retrieving a relevant document is double the cost of retrieving a non-relevant one (definition of T9U). Therefore, the thresholds should be lower than the midpoints to retrieve more relevant documents.

KUNa2T9U and KUNa2T9P use a decay for training documents with half life set to 2 years; we have found this value reasonable for filtering medical articles. Term selection cutoff is set at the top-500 terms; a light cutoff because our threshold optimization seems

[14] A brief description of query zoning is given in Appendix A.4

to require at least 250 terms in a classifier (Section 4.4.1), and moreover, long classifiers are necessary when tracking relevance drifts (Arampatzis and van der Weide, 2001). Thresholds are S-D optimized, however, not exactly as we have described in Section 4.5.

Our S-D method was in an early stage at the time of submission. What we did was to approximate the N_+ document scores with a Gaussian. Repeatedly adapting a query causes the distribution of non-relevant retrieved document scores to look more like a bell-shaped distribution. This is an artifact of re-training, however, and does not correspond to what is really happening below the threshold. Nevertheless, it has worked out reasonably, suggesting that a Gaussian approximation may be usable since it still gives some estimation of the *spread* of the non-relevant scores; however, it is of dubious accuracy. We will come back to this in Section 5.5.6.6.

For KUNbaT9U (batch-adaptive) we basically use the same settings as for KUNa2T9U, except for the Rocchio parameters. Moreover, we apply *document sampling* and *query zoning* (Singhal et al., 1997). The training stream (OHSUMED 1987) consists of around 54,000 documents, and only a few of them are relevant for a topic. For efficiency reasons we do random sampling with probability 0.1 to reduce the number of non-relevant training documents. Then we apply query zoning to select and use for training only the top-r scoring non-relevant documents, where r is the number of relevant training documents. We calculate the query zone with Equation 5.3 for $\gamma = 0$.

The adaptive runs do not show large differences in effectiveness, mainly because of the modest parameter settings for term selection cutoff, half life value, and the fact that the S-D threshold optimization technique is triggered only when at least 5 relevant and 5 non-relevant training documents are made available. Many topics did not reach these numbers, so they were actually filtered with thresholds set at weighted midpoints.

Next, we provide the extra runs we made in order to find where some of the parameters of FilterIt peak, and determine which techniques actually work. All runs reported here use (unless otherwise noted): query zoning to select for training only the top-r non-relevant documents, term selection cutoff set at 500, no decay, and thresholds set at weighted midpoints for T9U.

5.5.6.3 Document Sampling and Query Zoning

We have investigated the effect of sampling the non-relevant document space. We have run a batch-adaptive task with 3 different samples. Table 5.4 presents T9U and F-measure results. All samples are made by selecting randomly one out of ten non-relevant training documents from OHSUMED 1987. Then query zoning is applied before training the initial classifier. The results do not show significant differences.

sample	T9U	F_1
A (official)	19.5	0.406
B	19.8	0.406
C	19.1	0.403

Table 5.4: The effect of sampling the non-relevant training document space.

5.5.6.4 Term Selection

Figure 5.3 shows the impact of our term selection method (see Section 5.5.4) for different cutoff values. The runs are batch-adaptive using sample A. The average T9U seems to peak between 500 and 125 terms.

5.5.6.5 Decay

We have experimented with different half life values on an adaptive task. Figure 5.4 shows that the average T9U peaks somewhere between 2 and 8 years of half life. However, further analysis has revealed that effectiveness peaks at considerably different half life values across topics. An optimization of half life per topic — if we only had a way to do that — would have resulted in great improvements of the average T9U.

5.5.6.6 Threshold Optimization

In Appendix C.3 we give the TREC-9 evaluation table of our submitted batch and batch-adaptive runs. We have made a supplemental batch-adaptive run with the revised S-D threshold optimization as described in Section 4.5, i.e. by fitting an exponential on the top-50 non-relevant training documents[15]. When the non-relevant training document buffer exceeds 50 documents, we sort them according to their scores and discard the lowest scoring one. The results are presented in the last column, labeled as `FilterIt-ba`. They show an improvement in the average T9U from 19.4 to 21.3.

One could argue that setting thresholds with the weighted midpoint method works out comparably to the S-D optimization (compare e.g. KUNa1T9U to KUNa2T9U), but this is not the case. In fact, the good performance of the weighted midpoint method has been purely accidental; the same goes for the aforementioned Gaussian fit on non-relevant document scores. The mean score of non-relevant documents μ_{nr} has been estimated on the top-scoring non-relevant documents. This produces a relatively large μ_{nr}, which in its turn results in tight thresholding. When we have tried to increase the number of non-relevant documents, the weighted midpoint method as well as the Gaussian fit have greatly failed: the more non-relevant documents are used for training, the lower the μ_{nr}, thus lower thresholds. The methods fall too easily into the *selectivity trap* of retrieving too many (mostly non-relevant) documents. The revised S-D optimization, as described in Section 4.5, has proved much more reliable and robust in a range of settings, consistently avoiding such selectivity traps.

[15] Note that we have not optimized any other parameter according to our post-official runs; we have merely used a better S-D optimization.

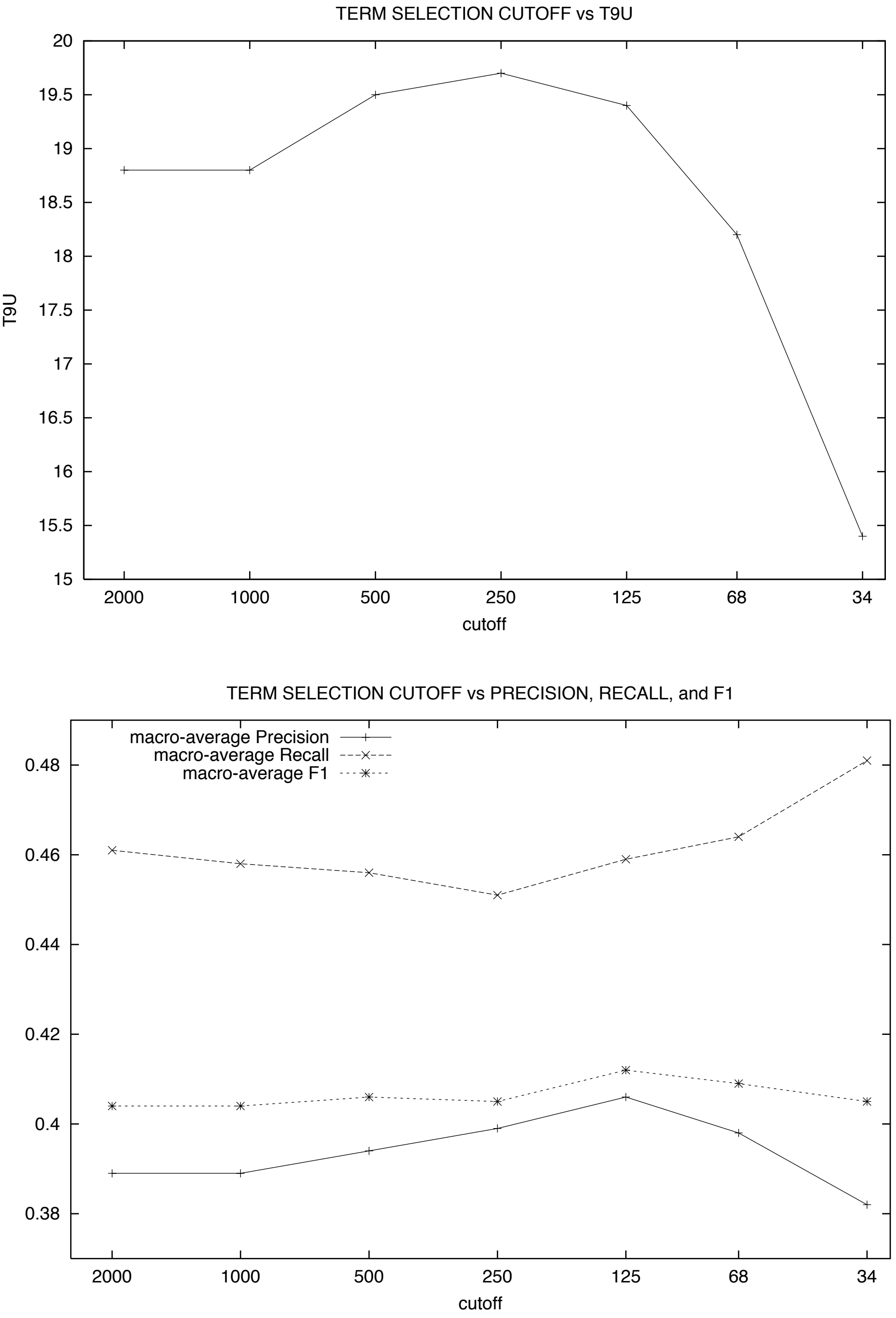
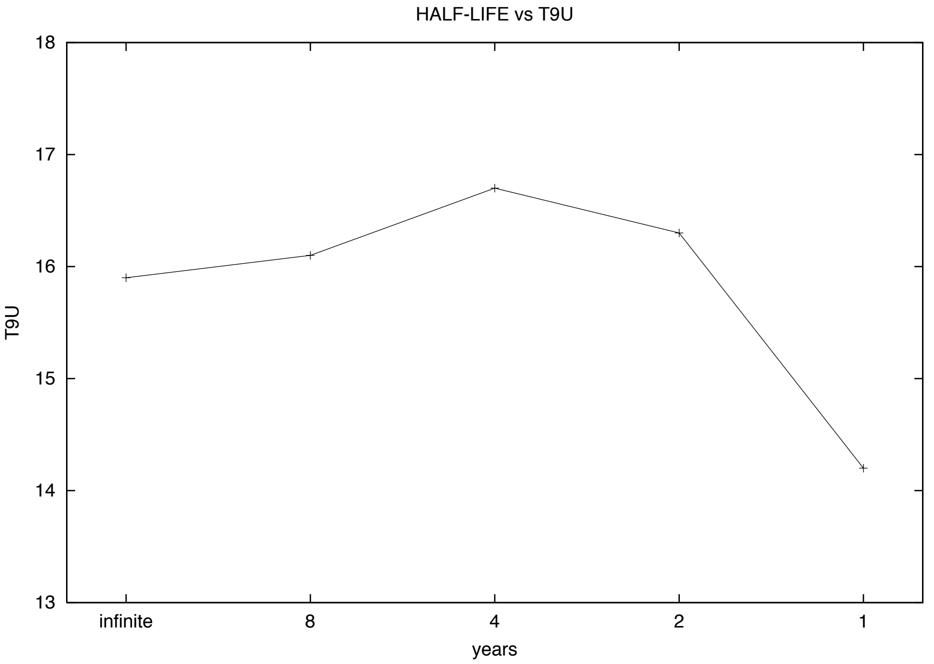


Figure 5.3: The effect of term selection.



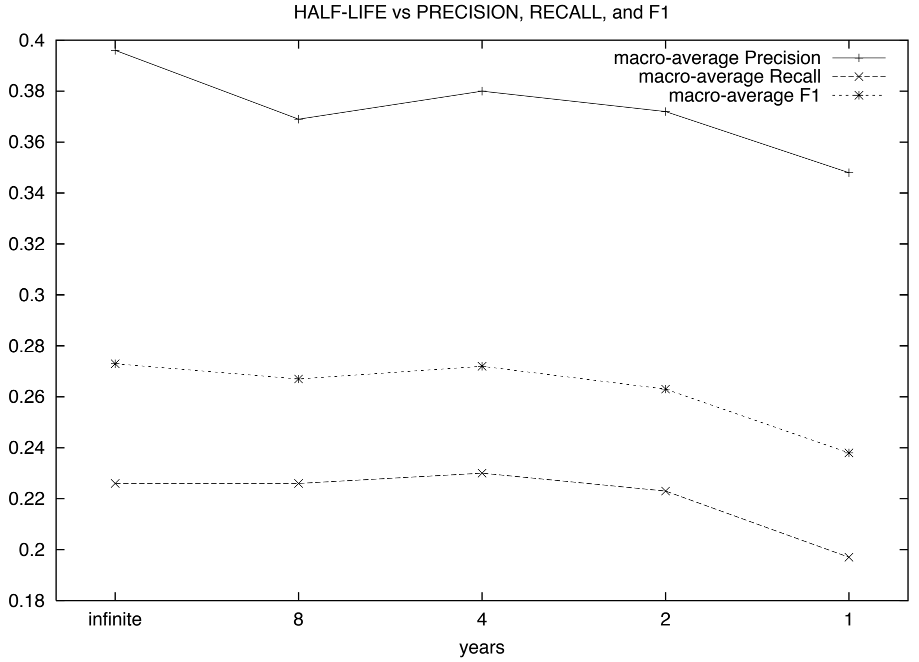


Figure 5.4: The effect of decay.

5.6 The LCS System

The routing and batch filtering tasks were carried out[16] by the LCS system[17] (Ragas and Koster, 1998). The system is based on the Winnow mistake-driven learning algorithm (Littlestone, 1988). The Winnow algorithm has, to our knowledge, not been used before in TREC. It can cope well with large numbers of terms, which is certainly the case here: after pre-processing, the training set had some 52,000 different terms.

5.6.1 The Winnow Algorithm and Improvements

During training, the *Balanced Winnow* algorithm (Littlestone, 1988; Dagan et al., 1997) iteratively computes two weights $w^+_{i,C}$ and $w^-_{i,C}$ for every term i and class (topic) C. These *winnow weights* are used to compute the score $S(D,C)$ of a document D for the class C as:

$$S(D,C) = \sum_{i \in D} (w^+_{i,C} - w^-_{i,C}) * u_{i,D} \ , \tag{5.11}$$

where $u_{i,D}$ is the *term strength* (weight) of term i in document D. Classification is achieved by thresholding $S(D,C)$ using a threshold θ.

Winnow is *mistake-driven* in the sense that it adjusts the weights $w^+_{i,C}$ and $w^-_{i,C}$ only if their current value, during an iteration, leads to a misclassification. If a relevant document scores below θ, then the winnow weights for the terms occurring in the document are multiplicatively updated using a promotion factor Alpha. Similarly, for a non-relevant document scoring above θ, the weights are demoted using a demotion factor Beta. The threshold θ is considered fixed, and the learning stops when there are no weight updates during an iteration, or earlier even in order to avoid over-training. Topic descriptions were considered as normal documents, since Winnow provides no special mechanism for dealing with requests.

The implementation of Winnow in LCS is similar to the one described in (Dagan et al., 1997), with two modifications:

1. the document terms $u_{i,D}$ are *ltc* weighted (Buckley et al., 1994), without the vector length normalization factor. Traditionally, $u_{i,D}$ are set either to the frequency of i within D, or to the square-root of the frequency. In experiments on the FT corpus, *ltc* has proved to work definitely better than the former, and slightly better than the latter.

2. Winnow weights were initialized for training as:

$$w^+_{i,C} = \frac{2\theta}{ADS} \ , \quad w^-_{i,C} = \frac{\theta}{ADS} \ , \quad ADS = AVG_D \frac{\sum_{i \in D} u_{i,D}}{size(D)} \ ,$$

   where $size(D)$ is the number of unique terms in document D. This initialization improves Winnow's convergence speed.

[16] All methods and experiments described in Section 5.6 are attributed to my colleague Jean Beney, who joined KUN during his sabbatical leave from INSA de Lyon.

[17] Esprit project DOcument ROuting (DORO): `http://www.cs.kun.nl/doro`

The convergence speed of the Winnow algorithm (the number of iterations needed to learn a stable classifier) depends rather critically on the initial values of the weights. In (Dagan et al., 1997), all positive weights are initialized as θ/d, where θ is the threshold and d the average number of "active features" in documents. This choice ignores collection statistics for terms. In our initialization, an average document obtains an initial score equal to θ. Since term strengths are taken into account, fewer iterations are needed.

5.6.2 Threshold Setting by Cross-evaluation

The Winnow algorithm has a "natural" threshold $\theta = 1.0$ for separating relevant from non-relevant documents, giving rather equal utility to retrieving a relevant document and rejecting a non-relevant one. T9U stresses relevant documents more than non-relevant, however. The S-D threshold optimization has not (yet) been implemented in the LCS, so the necessary threshold optimization was performed by *cross-evaluation.*

The training set (OHSUMED 1987) was split into n subsets of the same size, which each in turn was used as *optimization test set* while all the other subsets, together with the topic descriptions, were used as *optimization training set*. The scrap of the split was included into the optimization test set. After training Winnow with $n - 1$ subsets, the documents of the remaining subset (optimization test set) were ranked according to their scores. Then, by going down the rank, the threshold value that optimized T9U was found. We performed the cross-evaluation for $n = 2$, 3 and 4, and we took the mean of all $(2 + 3 + 4 = 9)$ optimal threshold values.

5.6.3 Experiments with LCS

5.6.3.1 Submitted Runs

We set the Winnow parameters to the values that gave the best results on the FT corpus (Table 5.5). We use the *thick separator* heuristic (Dagan et al., 1997): instead of a single threshold θ, a *threshold range* $[\theta^- : \theta^+]$ is used. There is a promotion whenever a relevant document obtains a score below θ^+ and a demotion when a non relevant document gets a score over θ^-. This heuristic achieves a better separation between relevant and non-relevant documents. The asymmetry around the standard threshold (1.0) forces the algorithm to perform more promotions than demotions on the early iterations. This compensates for the asymmetry between the numbers of relevant and non-relevant training documents, speeding up convergence.

parameter	value
Alpha	1.1
Beta	0.9
ThresholdRange	on
θ^+	1.3
θ^-	0.9
MaxIters	30

Table 5.5: Winnow Parameters.

We have submitted 2 routing runs, KUNr1 and KUNr2. LCS has originally been developed for *mono-classification* tasks, i.e., each document belongs to exactly one class. This means that the relevant training documents for one class are considered as non-relevant training documents for all other classes. That is certainly not the case in filtering, so we had to do separate runs per topic assuming two classes: relevant and non-relevant. The routing results KUNr1 were produced like this.

The approach of separate runs is correct but obviously inefficient. So, we also tried to process all topics at once (KUNr2), hoping that they do not have relevant documents in common, or even if they do, the impact of this approach on effectiveness would not be that great. Luckily, in the given dataset, it was not: the average uninterpolated precision was practically the same. We obtained 0.237 for KUNr1 and 0.234 for KUNr2.

The batch filtering run KUNb was obtained through the thresholding of the rankings of KUNr1. Thresholding was performed by the cross-evaluation method described in Section 5.6.2.

5.6.3.2 Other Runs

The KUNb results, obtained with separate thresholds per topic calculated by cross-evaluation, can be compared with those obtained by a simpler method: a uniform threshold for all topics. We can choose as a uniform threshold any value in the threshold range; such a choice should give the same result if the classification is perfect. But two values are special: 1.0 (average document score before training), and 1.1 (the center of the threshold range).

Table 5.6 shows that the results for $\theta = 1.0$ are worse than those for $\theta = 1.1$. Moreover, a uniform threshold set at 1.1 gives slightly better results than the separate thresholds computed by cross-evaluation. It seems that the cross-evaluation method has failed, mainly because the training sets had relatively small numbers of relevant training documents. Splitting the sets for cross-evaluation, made things even worse.

Run	T9U
separate θ's via cross-evaluation (KUNb)	5.0
uniform $\theta = 1.0$	−3.5
uniform $\theta = 1.1$	6.0
best possible thresholdings on KUNr1	17.9

Table 5.6: Different thresholdings on Winnow.

The best possible thresholdings on the rankings of KUNr1 would have obtained an average T9U of 17.9; not very good either, considering that the largest possible average T9U for the given test set is 104.9. This implies that the rankings achieved are not very good. Winnow does not perform well for small numbers of relevant training documents.

5.7 Overall Comparison and Discussion

In Appendix C.4 we give the TREC-9 evaluation table of our submitted routing runs with the LCS system. The right-most column, `FilterIt-r`, corresponds to a supplemental routing run with the FilterIt system. Obviously, FilterIt gives better rankings than LCS; the corresponding average uninterpolated precision figures are 0.373 and 0.237. Thresholding the rankings of `FilterIt-r` with the optimal S-D thresholds (as these were estimated by the method described in Section 4.5.3) we obtained a (non-adaptive) batch run with FilterIt. Its results are presented under the label `FilterIt-b` in Appendix C.3. An average T9U of 14.8 is obtained in contrast to 5.0 obtained by LCS.

Since `FilterIt-r` and `FilterIt-b` are not *post-factum* optimized, it seems that we should have submitted all runs, for all filtering tasks, with the FilterIt system. The `FilterIt-r` routing run, with an average precision of 0.373, would have ranked us as second best system; the first system scored at 0.385. The `FilterIt-b` batch run, with an average T9U of 14.8, would have ranked us clearly as the best system; the best official batch run scored at 7.5. All official TREC-9 runs of the tasks we have participated are given in Tables 5.7 and 5.8.

Adaptive filtering — OHSU topics			
T9U	Run	T9P	Run
17.349	KUNa2T9U	0.294	ok9f1po
16.762	KUNa1T9U	0.288	ok9f2po
10.746	ok9f3uo	0.279	CMUDIR16
10.095	CMUDIR17	0.267	CMUDIR14
9.698	ok9f1uo	0.265	FDUT9AF3
9.556	FDUT9AF2	0.264	FDUT9AF1
9.270	CMUDIR15	0.258	KUNa1T9P
1.143	reliefs1	0.249	FDUT9AF4
-5.873	IOWAF001	0.230	KUNa2T9P
-32.270	antadapt001	0.224	CMUCAT5
-35.302	kddaf903	0.213	CMUCAT3
-35.492	kddaf905	0.168	reliefs2
-35.857	kddaf906	0.138	IOWAF003
-36.381	kddaf904	0.102	antadapt002
-43.571	antadapt002	0.088	antadapt001
-55.683	pircT9U1	—	—
-69.143	pircT9U2	—	—

Table 5.7: TREC-9 adaptive filtering runs with OHSU topics.

At any rate, we are very satisfied with the performance of the FilterIt system in our official runs. We have clearly achieved the best scores in all adaptive and batch-adaptive tasks optimized for T9U. Compare the 17.3 (KUNa2T9U) and 19.4 (KUNbaT9U) to the 10.7 and 13.6 of the second best systems in the corresponding tasks. The official T9P runs are also satisfactory; our best run has achieved 0.258 (KUNa1T9P), a rather comparable effectiveness to the 0.294 of the best system. After all, we have not optimized exactly for T9P, but for some other related utility measure, in order to simplify the calculations (see Section 5.5.5).

Batch filtering — OHSU topics (∗ indicates batch-adaptive)		Routing — OHSU topics	
T9U	Run	Av. Prec.	Run
19.365	KUNbaT9U∗	0.385	S2RNr2
13.587	FDUT9BF1∗	0.343	S2RNr1
7.460	mer9b1	0.326	ok9rf2po
5.048	KUNb	0.317	ok9rfr2po
2.778	scai00	0.237	KUNr1
1.095	mer9b2	0.235	mer9r1
—	—	0.234	KUNr2
—	—	0.185	Mer9r2
—	—	0.177	antrpnohsu00
—	—	0.099	antrpohsu00
—	—	0.081	lsir1

Table 5.8: TREC-9 batch filtering and routing runs with OHSU topics.

Why are the results with the LCS less satisfactory? According to our experience, Winnow performs better than Rocchio when large numbers (hundreds) of relevant training documents are available for each class. This was not the case in the batch and routing tasks of TREC-9 where some topics had very few relevant training documents. This may largely be responsible for Winnow's weak performance. Furthermore, with 30 iterations in the learning phase, there is some evidence of overtraining.

Why are the results with FilterIt so good? Let us summarize the methods we have used: accurate and incremental adaptivity as soon as a single training document becomes available (in contrast to re-training in batches), local adaptivity (training documents of decaying value in time), on-the-fly term selection (in contrast to just cutting off classifiers), the S-D threshold optimization (note that we are talking about "optimization" rather than "setting"), and initial query elimination. Moreover, all parameter settings (e.g. Rocchio's α, β, γ, term selection cutoff, half life) have either been empirically determined on the Financial Times collection or at least motivated. There is evidence as well that *Ltu* weighting and query zoning have contributed considerably to effectiveness. The FilterIt system is a typical example of: *the whole is more than the sum of its parts*.

5.8 Summary

In this first-time contribution to TREC, we have focussed mainly on the adaptive tasks. Our contribution to adaptive filtering has been threefold:

- We have investigated the value of retrieved documents as training examples in relation to their time of retrieval. For this purpose, we have introduced the notion of the *half-life* of a training document. The approach has presented promising results.

- We have put in practice the *score-distributional (S-D) threshold optimization* method, capable of optimizing any effectiveness measure defined in terms of the traditional contingency table. The method has found to be very effective, and it can moreover be applied incrementally.

- We have developed a system that allows *incremental adaptivity*, minimizing its computational and memory requirements without sacrificing too much accuracy.

Overall, we are very satisfied with our adaptive results; we have clearly achieved the best utility scores in all adaptive and batch-adaptive tasks that we have participated in. The results of the batch and routing tasks are less satisfactory, but at least the feasibility of using the Winnow algorithm in these applications has been demonstrated.

Summarizing, our TREC-9 participation has motivated a great deal of research. As a result, we have finalized the S-D threshold optimization as described in Chapter 4, and we have re-considered the nature of the filtering task as described in Chapter 2. Our plans for further research include: finding a way of detecting relevance drifts in order to select appropriate half life values, and to revise the term selection method we have introduced in Chapter 3.

Chapter 6

Linguistically Motivated Indexing

This chapter summarizes our theoretical work in the area of using natural language resources and processors for information seeking tasks. Although our main concern is information filtering, the ideas apply to all information seeking tasks which involve indexing of textual information objects. Therefore, we discuss the issues assuming an information retrieval context. The chapter is based on our previously published work in (Arampatzis et al., 1998) and (Arampatzis et al., 2000b).

6.1 Introduction

Information retrieval (IR) has been developed to provide practical solutions to people's need to find the desired information in large collections of data. The IR task can be seen as the "digital twin" of the task of a person looking in a library for material relevant to a certain subject. In both cases, the searcher has an *information request* that has to be translated to library indices or *query* terms. Then it is submitted to some system — library catalogue or computerized retrieval system — and the system in turn suggests (retrieves) relevant material. The searcher will usually find that some of the suggested documents are not actually relevant, and will also suspect that some relevant documents might have been missed. For static collections, the effectiveness of such a search can be quantified using two metrics, *precision* and *recall* (Appendix A.5). For an extended introduction to the IR problem, its history, widely accepted techniques, and retrieval evaluation metrics, the reader should refer to the classical books (Salton and McGill, 1983) and (van Rijsbergen, 1979); for a collection of classical articles in IR, to (Jones and Willett, 1997).

The tremendous increase over the last decade in information in digital form has led to a new challenge in IR. A World Wide Web search today retrieves a large number of hits, and usually imposes the tedious task of going through hundreds of irrelevant hits on the user. Although IR has been in existence for more than three decades (and as a part of library science even longer), modern technology for its part is still based on a simple assumption that often leads to unsatisfactory results. Restricting the problem to textual data, the assumption, implicit or explicit, upon which most commercial IR systems are based, is that

Definition 1 (naive keyword retrieval hypothesis) *If a query and a document have a (key)word in common, then the document is to some extent about the query.*

Of course, if they have *more* keywords in common, then the document is *more* about the query. Moreover, the keywords are usually augmented with weights indicating their importance as information discriminators. In this respect, the IR problem is represented by matching the "bag" of keywords in the user's query with the bag of keywords representing the documents. The output of such a matching is usually a *ranked* list of documents with the most relevant first and the least relevant last.

This relatively simple representation is the computer age equivalent of library catalogues, and carries the same inadequacies. The most obvious inadequacies originate from *linguistic variation*, making the keyword retrieval hypothesis insufficient because

1. it does not deal with *morphological variation* which produces keywords in singular and plural form, for instance *wolf* and *wolves*, or different cases, such as *man* and *man's*. (Dealing with cases is trivial for English, but it is crucial for other more inflected languages like German or Greek).

2. it does not handle cases where different words are used to represent the same meaning. For this phenomenon we use the term *lexical variation*. The result is that a query with the keyword *film* does not retrieve documents that contain its synonym, *movie*.

3. it does not distinguish cases where single words have multiple meanings due to *semantic variation*. A singer looking for *bands* will be faced with *radio frequency bands* as well.

4. it does not deal sufficiently with *syntax*. The problem is twofold.

   (a) *accidental co-occurrence*: A document that contains the phrase *near to the river, air pollution is a major problem* is not about *river pollution*, although both keywords occur in the document, and certainly *science library* is not the same as *library science*.

   (b) *syntactic variation*: This problem shows up in retrieval models which use exact phrase matching, rather than in keyword-based models. For example, *polluting the river* and *pollution of the river* are both about *river pollution*, but looking only for the literal occurrence of the latter phrase will miss documents containing any of the first two phrases.

   Both problems of accidental co-occurrence and syntactic variation can be dealt with by taking into account the syntax.

Linguistic variation degrades the effectiveness of IR systems in terms of precision and recall. On the one hand, morphological and lexical variation hurts recall. On the other hand, semantic and syntactic variation hurts precision. However, trying to improve recall usually decreases precision and vice versa.

Linguistic variation in the IR context may be interpreted as meaning that language is not merely a bag of words. Language is a means to communicate about concepts, entities, and relations, which may be expressed in many forms. Word order may matter (as in *science library* vs. *library science*) or may not (*general director* vs. *director general*). Moreover, words combine to form phrases and other larger units with a meaning that may not be directly inheritable from the individual words. For example, a *hot dog*, either hot or not, has nothing to do with dogs. Given such considerations, it has been conjectured many times that a better representation should also include groups of words (phrases) and some form of regularization of words, word order, and meaning. Indeed, many researchers have developed such techniques.

This chapter discusses a retrieval schema that attempts to overcome some of the problems originating from the keyword retrieval hypothesis and linguistic variation. In the next section, we will review some of the most important attempts made to deal with linguistic variation. In the rest of the chapter, we will discuss the key aspects of a linguistically motivated retrieval system. Starting in Section 6.3 from a phrase retrieval hypothesis — a naive extension of the keyword retrieval hypothesis — we will address a suitable representation of phrases for IR in Section 6.4. In Section 6.5, possible regularizations of natural language will be outlined. The weighting of phrasal indexing terms and their matching will be discussed in Section 6.6. An example architecture of such a linguistically motivated retrieval system will be depicted in Section 6.7. We will draw some conclusions in the final section.

6.2 Related Research

The problems of linguistic variation have been noted by many researchers, who have answered with various techniques. Many of these techniques employ natural language processing (NLP) and such language resources as online dictionaries and thesauri. The results until now have been inconsistent, making it difficult to reach a conclusion about their effectiveness.

In this section, we review some approaches and their outcomes for each of the morphological, lexical, semantic, and syntactic variation. Special attention is given to three studies which we consider representative and closely related to the approach we are going to take in this chapter. These studies are the works of the IR group at Dublin City University, the Clarit group, and Strzalkowski et. al. (Strzalkowski and Carballo, 1995; Strzalkowski et al., 1997).

6.2.1 Morphological Variation

Morphology is the area of linguistics concerned with the internal structure of words. It is usually broken down to two types, *inflectional* and *derivational*. Inflectional morphology describes the predictable changes a word undergoes as a result of syntax, and has no effect on the word's part of speech (e.g., a noun remains a noun) and little effect on its meaning. The most common changes are the plural and possessive forms of nouns (e.g., *computer, computers, computer's*), comparative and superlative form of adjectives (e.g., *good, better, best*), and the past tense, past participle, and progressive form of verbs (e.g., *compute, computed, computing*). On the contrary, derivational morphology may or may not affect part of speech or meaning (e.g., *computerize, computerization*).

Two ways have generally been followed to deal with morphology in IR trying to increase recall. These are *variant query expansion* and *stemming*. In variant query expansion, morphological variants of keywords are added to the query. Stemming simply strips a word's suffix to reduce it to its *stem*, assuming that keywords with a common stem usually have similar meanings. Variant query expansion and stemming can been regarded as equivalent and the choice depends on the nature of the particular application. We will concentrate on stemming as the choice that is made the most.

Stemming can be done in a linguistic fashion, taking into account the function and the part of speech of a word, or in a nonlinguistic fashion, disregarding a word's context. Lovins and Porter developed nonlinguistic algorithms for suffix stripping based on a list of frequent suffixes to reduce words to their stems (Lovins, 1968; Porter, 1980). It is a common belief that stemmers improve recall without losing too much precision, however, a comparison of the Lovins stemmer, the S stemmer, and the Porter stemmer with a baseline of no stemming at all, concluded after detailed evaluation that none of the three stemming algorithms consistently improves retrieval for English documents (Harman, 1991). It was argued that the evaluation measures were not appropriate, and new measures were proposed for evaluating the performance of different stemming algorithms (Hull, 1996). After experimentation, it was concluded that stemming is almost always beneficial for English, except for long queries at low recall levels. A more reliable version of Porter's stemmer was developed, which uses a dictionary to validate the result after every suffix-stripping step. This revised Porter stemmer resulted in improvements in retrieval performance for English documents, especially short ones (Krovetz, 1993).

Research with other morphologically more complex languages such as Slovene showed an improvement in effectiveness using a Porter-like stemmer modified for Slovene (Popovic and Willett, 1992). In the same study, when the Slovene corpus was translated to English and the experiment was repeated, there was no improvement in retrieval. For Dutch texts, it was found that linguistic inflectional stemming improves recall without significant loss in precision, while derivational stemming, although sometimes useful, in general reduces precision too much (Kraaij and Pohlmann, 1996b).

6.2.2 Lexical Variation

Lexical variation has generally been treated in two ways. On the one hand, by *lexical query expansion* with semantically related terms (e.g., synonyms), and on the other hand, the matching of query and document keywords via *conceptual distance measures*. For these purposes, thesauri have been exploited to supply related query terms, and semantic networks such as that of WordNet (Miller, 1995) to define semantic distance measures between words.

The choice of semantically related terms for a word depends on the context in which the word is used; thus, the context specifies the word's *sense*. When a word can be used in different senses, the problem of *word sense ambiguity* arises. Most of the techniques that deal with lexical variation require prior word sense disambiguation, and that makes these techniques strongly dependent on semantic variation (described in the next section).

Query expansion with WordNet has shown a potential in enhancing recall since it permits the matching of relevant documents that do not contain any of the query terms (Smeaton et al., 1995). Expansion of queries using synonymy and other semantic relations supported by WordNet showed that short and incomplete queries can be significantly improved, yielding better retrieval effectiveness (Voorhees, 1994). However, this query expansion technique made little difference in the effectiveness, for relatively complete descriptions of the information sought. For Dutch texts, synonym expansion was reported as potentially useful (Kraaij and Pohlmann, 1996a).

Experiments on a small collection of image captions (i.e., very short documents) using measures of semantic similarity distance between words based on WordNet showed improvements in retrieval (Smeaton and Quigley, 1996). However, their earlier experiments with word-to-word semantic similarity measures resulted in a drop in effectiveness, due to the effects of erroneous word sense disambiguation (Richardson and Smeaton, 1995).

Another approach, based on indexing in terms of WordNet's synonym sets (synsets) instead of wordforms, yielded successful results when queries were fully disambiguated (Gonzalo et al., 1998). If queries are not disambiguated, indexing by synsets at best performs only as well as standard word indexing.

6.2.3 Semantic Variation

Semantic variation has strong impacts on lexical query expansion, on matching based on word-to-word semantic distance similarity measures, and on conceptual indexing. The success of these techniques requires prior disambiguation of word senses, as many researchers have noted (Voorhees, 1994; Kraaij and Pohlmann, 1996a; Smeaton and Quigley, 1996; Gonzalo et al., 1998). Most of the research has concentrated on how large the impact of semantic variation and its inaccurate resolution is on IR effectiveness.

It is estimated that if word sense disambiguation is performed with less than 90% accuracy the retrieval results are worse than not disambiguating at all (Sanderson, 1994). Poor retrieval results have been blamed on this reason in previous research (Richardson and Smeaton, 1995). Conversely, in the same experiments (Sanderson, 1994) word sense ambiguity was shown to produce only minor effects on retrieval accuracy, apparently suggesting that query–document matching strategies already perform an implicit disambiguation. In this experimental setup, ambiguity was introduced artificially by substituting randomly selected word pairs such as *bank* and *spring* with ambiguous terms like *bank/spring*. This setup has two disadvantages, first, real ambiguity might not behave like the artificially introduced one, and second, the disambiguation of an artificially ambiguous term is only partial; when *bank/spring* is disambiguated as *bank*, *bank* is still ambiguous as it can be used in more than one sense in a text collection (Gonzalo et al., 1998).

6.2.4 Syntactic Variation

The techniques developed to deal with syntactic variation may be grouped in two categories: the addition of *phrases* to queries, and the use of *syntactical structures* for indexing. These techniques intend to increase retrieval precision.

A phrase is a group of words, and historically what has been referred to as a phrase in the IR context varies significantly among researchers. The hypothesis for using phrases has been that they denote more meaningful entities or concepts than single words; thus they may constitute a better representation. Indeed, the use of phrases has become common in IR; many systems participating in the Text REtrieval Conferences (TRECs) now use one or another form of phrase extraction (Voorhees and Harman, 1997).

Traditionally, two types of phrases have been used, *statistical* and *syntactic*. Statistical phrases are any series of words that frequently occur contiguously in a text collection. Syntactic phrases are any set of words that satisfy certain syntactic relations or constitute specified syntactic structures. Statistical phrases are extracted using word frequency and co-occurrence information, while syntactic phrases usually require sophisticated NLP techniques. Which of the two types is more useful for IR remains unclear; syntactic phrases seem to offer an advantage that is statistically rather insignificant (Fagan, 1987; Mitra et al., 1997; Kraaij and Pohlmann, 1998). Small statistically insignificant improvements were found for Dutch texts (Kraaij and Pohlmann, 1998). Other research concluded that phrases do not have a major effect in precision at high ranks, but are more useful at lower ranks (Mitra et al., 1997).

6.2.5 The IR work at Dublin City University

The IR group at Dublin City University tried the use of indexing structures derived from syntax. We review the approach and results from their participation in TREC-3 (Smeaton et al., 1994), since that was their last attempt to use syntactic phrases.

In their approach, documents and queries were represented by TSAs (tree structure analytics) constructed at the clause level. These TSAs were directly derivable from a morpho-syntactic analysis of input text, and were formulated to encode within their structures the most commonly occurring syntactic ambiguities due to prepositional phrase attachment, conjunction, and others. In case of ambiguity, the TSA matching algorithm weights various (syntactic) interpretations at the time of retrieval. This TSA matching algorithm is able to measure the degree of overlap between input phrases which may or may not be about the same topic, but which use the same words albeit sometimes in different contexts. The degree of overlap is inferred from the structure roles different words play in phrases, acting as heads, as modifiers or as attachments.

The group conducted an experiment on category B of TREC-3 (i.e. on 550 Mbytes of the Wall Street Journal), and reported failure. The implementation was based on a two-stage retrieval. Firstly, a statistically-based pre-fetch retrieval ranked the collection. Then the computationally expensive language-based processing was applied to the 1000 top-ranked documents in order to re-rank them.

The experimental results were disappointing and unexpected (both recall and precision were decreased). The group posed some possible reasons for the poor results:

- The language analyzer used was of poor quality.

- The type of language used in TREC topic descriptions is very different to that used in document texts (interrogative vs descriptive language), and the two types of language should have been treated differently.

- Maybe the combination of independent retrieval strategies (pre-fetch using *tf* × *idf* and TSA-based weighting in this case) would have bootstrapped the performance of individual strategies (this has been shown before by a number of groups in TREC-3 and elsewhere). Maybe the TSA-based retrieval could have retrieved documents that were not retrieved by the term weighting strategy, especially if these documents had a few words in common with a query but these words played the same or similar structural roles in the query and in the document.

The results led the group to conclude that the approach of using syntax to determine structural relationships between words and to use them as a part of an information retrieval strategy, does not work. Since then, the group has abandoned this strategy and it concentrated on the use of NLP resources (such as machine-readable dictionaries and knowledge bases) to improve retrieval.

6.2.6 The Clarit work

The Clarit system has several NLP techniques integrated with the vector space retrieval model (Zhai et al., 1996). These techniques include morphological analysis, robust noun-phrase parsing, and automatic construction of thesauri. Clarit's indexing emphasizes phrase-based indexing with different options for decomposing noun phrases into smaller constituents, including single words.

The goal of the Clarit TREC-5 NLP experiment was to test two hypotheses:

1. The use of lexical atoms, such as "hot dog", to replace single words for indexing would increase both precision and recall.

2. The use of syntactic phrases, such as "junior college" to supplement single words would increase precision without hurting recall, and using more such phrases results in greater improvement in precision.

For the first hypothesis, lexical atoms were considered the high frequency word pairs that tended not to be separated by other words within the context of noun phrases. The only pairs considered were: two nouns, or one adjective followed by a noun. In both TREC-5 and the preliminary experiments with TREC-4 topics, it was shown that the use of lexical atoms leads to a slight but consistent improvement in average precision. On the other hand, the use of lexical atoms did not consistently improve recall and initial precision. In fact, it increased either recall or the initial precision. The inconsistent influence of lexical atoms may indicate a need for better control over the selection of phrases that are used for replacing single words.

For the second hypothesis, syntactic phrases were obtained from noun phrases (NPs). The noun phrase parser used an expectation maximization algorithm to obtain statistical evidence of word modifications from the noun phrases in the corpus (Zhai, 1997). In other words, they applied statistical methods to assign structure to those noun phrases which had an ambiguous structure (all noun phrases of more than two words). The three automatic official runs of the experiment corresponded to the following three levels of term combinations: *a)* single words only, *b)* single words + head modifier pairs + full

NPs, and *c)* single words + head modifier pairs + adjacent sub-phrases + full NPs. These experiments in supplementing single words by various combination of syntactic phrases in the indexing process showed a consistent and significant improvement in retrieval performance. However, the impact of adding phrases into the index space varied according to the query topic. Thus, while adding phrases helped some topics it hurt some others.

6.2.7 Natural Language Information Retrieval in TREC-4

The approach of (Strzalkowski and Carballo, 1995) in TREC-4 was rather successful. They built an NLP module around a statistical full-text indexing and search backbone. The NLP module was used to *a)* extract content-carrying phrases from documents, and *b)* process user's natural language requests into effective search queries.

All TREC-4 texts were processed with a syntactic parser. Phrases were extracted from the parse trees and used as compound indexing terms in addition to single keywords. Statistical methods were applied to resolve structural ambiguity. These phrases were head-modifier pairs.

The user's natural language request was also parsed to identify indexing terms. Highly ambiguous, usually single-word terms were dropped, provided that they also occurred in compound terms. Additionally, similarity relations, such as synonymy, hypernymy, hyponymy, etc., were considered for query expansion. For example, "unlawful activity" was added to a query containing the compound term "illegal activity", via a synonymy link between "illegal" and "unlawful".

Two types of morphological normalization were performed: *a)* inflected word-forms were reduced to their root forms as specified in the dictionary, and *b)* nominalized noun forms were converted to the root forms of corresponding verbs (e.g. "implementation" was converted to "implement").

The experiments showed a substantial improvement in precision when phrasal terms were used. A sharp increase in precision was achieved near the top of the ranking, which brings further gains in performance via automatic relevance feedback. The researchers cautiously suggested that NLP can be effective in creating appropriate queries out of user's natural language requests which can frequently be imprecise or vague. However, the benefit from linguistic processing was tied to the length of queries: the longer a query, the larger the improvement.

In subsequent TRECs, the group elaborated further on the techniques described here, but NLP has not yet been proven to be as effective as they would have have hoped in obtaining better indexing and representation of queries. Using linguistic terms does help to improve precision, nevertheless, the gains remain quite modest (Strzalkowski et al., 1997).

6.2.8 Beyond the Bag-of-words Paradigm

Various attempts have been made to break out of the bag-of-words paradigm. Experiments have shown considerable variation in retrieval effectiveness, making it difficult to establish which techniques actually work and which do not. Summarizing:

- The effectiveness of stemming depends on the morphological complexity of a language. Restricting the problem to English, there is a lot of variation in the results of stemming experiments, and a number of factors seem to be of importance, e.g., linguistic vs. nonlinguistic stemming, stemming algorithm, query and document length, and even evaluation measures.

- Lexical and semantic variation are strongly connected. It seems that dealing with lexical variation is more beneficial for incomplete and relatively short queries. Whether conceptual distance matching scales up to longer documents and queries is still an unanswered question. Moreover, most of the relevant research has shown that the successful application of these techniques is very sensitive to word-sense ambiguity. However, word sense disambiguation techniques are still not well established.

- It is still not clear how syntactic information can be used to improve retrieval effectiveness consistently. Questions still remain about which phrases are useful, in which cases, and how these should be extracted. Furthermore, NLP is still nowhere near to becoming practical in dealing with large amounts of textual data of unrestricted domain. Due to its lack of robustness and efficiency, compromises have to be made. NLP techniques have mostly been used to *add* indexing terms to a bag-of-words representation, and therefore trying to sharpen a keyword-based search. In this way, the inadequacies of NLP have been softened; in the worst case, a system will fall back on the original bag-of-words representation.

Although a lot of effort has been put into linguistically motivated retrieval schemes, whether or not this is worth the trouble remains unclear. The evidence suggests the need for further investigation and better modeling. In the rest of this study, we will describe a retrieval scheme that demonstrates the application of linguistically motivated techniques.

6.3 The Phrase Retrieval Hypothesis

The goal of the *indexing* task is to assign characterizations (*terms*) to documents that are deemed to best represent their content. Terms are usually derived from document content. Every term used to characterize documents of the same collection can be seen as adding a new dimensionality to the characterization. Terms should be assigned to documents in such a way that documents on the same topic are positioned close together in the hyper-dimensional document space, while those on different topics are placed sufficiently apart. Terms can be anything from tri-grams and words, for example, to linguistic-entities and concepts. In the two extreme cases, documents can be characterized by themselves (e.g., their document numbers), or all documents by exactly the same characterization. The former characterization positions documents as far apart as possible, resulting in no way of retrieving documents on the same topic. It is thus unusable in the IR context. The latter provides no way of discriminating between different topics. A suitable characterization must be *usable* and *discriminating.*

In a keyword-based representation, every document is characterized by a set of keywords with weights representing the importance of each keyword in characterizing the document. Keywords are usually derived directly from the document's text. Keyword-based representations are modestly usable and discriminating. Single words are rarely specific enough for accurate representation (e.g., the word *system* does not say much, whereas a *HiFi sound system* clarifies the meaning somewhat more). Moreover, a word with a high frequency of occurrence in a document collection is not a good discriminator. On the other hand, a phrase, even made up of high-frequency words, may occur in only a few documents, thus becoming a good discriminator. These observations suggest that a better characterization will make use of phrases; consequently, a *naive* phrase retrieval hypothesis can be formalized as follows:

Definition 2 (naive phrase retrieval hypothesis) *If a query and a document have a phrase in common, then the document is to some extent about the query.*

The phrase retrieval hypothesis does not solve the problems originating from the keyword retrieval hypothesis and linguistic variation. On the contrary, it creates more questions, such as what a phrase is and how it should be used for indexing or be weighted and matched. We use this definition merely as a starting point, upon which we will build our framework.

Phrases can be obtained using statistical or syntactic methods. Syntactic phrases appear to be reasonable indicators of content, arguably better than proximity-based statistical phrases, since they account for changes in the word-order[1] or other structural constructions (e.g., *science library* vs. *library science* vs. *library of science*). Experiments have shown, however, that syntactic methods are not significantly more effective than statistical methods (Fagan, 1987; Mitra et al., 1997; Kraaij and Pohlmann, 1998). This failure of NLP to outperform statistics can be attributed to the poor quality and robustness of the existing NLP techniques. Nevertheless, we will adopt a syntactic approach for the time being, assuming that accurate syntactic analysis and disambiguation techniques will become available. We will return to the effectiveness issues of NLP in Section 6.7.

Evidence suggests that noun phrases should be considered as a semantic unit. The most important reasons are

- noun phrases play a central role in the syntactic description of *all* natural languages, functioning as subject and object, and in preposition phrases.

- In artificial intelligence, noun phrases are considered as references to (or descriptions of) complicated concepts (Winograd, 1983). By others, as *picture producers.*

[1] Most approaches to statistical phrases do use word-order but they do not account for syntax; they either use a standard word-order (e.g. alphabetical) or the word-order as it is found in text.

Noun phrases might be good approximations of concepts, but other phrases also corresponding to concepts are missed. This observation points to the necessity to consider other phrases as well (e.g., verb phrases). The verb phrase describes a situation or process by relating a main verb to a number of noun phrases and other phrases. The linguistically meaningful phrases that may be considered as retrieval terms are therefore at least the noun phrase including its modifiers, and the verb phrase including its subject, object, and other complements. An abstract representation of these phrases suitable for indexing is needed, and will be defined in Section 6.4.

Phrases can be used in their literal form as terms, although the performance is then expected to be inferior to that of keywords. It is well known that as the size of a corpus grows, the number of keywords grows with the square root of the size of the corpus. One could expect that the same holds for phrases, but the number of such enriched terms grows even faster, as does the likelihood of there being different phrases corresponding to the same concept. On the one hand, we would like to use phrases to achieve precision, but on the other hand, recall will be too low because the probability of a phrase reoccurring literally is too low. To deal with this *sparsity* of phrasal terms, we shall introduce a number of *linguistic normalizations* (Section 6.5). Linguistic normalization tries to reduce alternative formulations of meaning to a *normalized form*. For example, *river pollution* and *pollution of rivers* are both normalized to the same indexing term `pollution+river`.

6.4 Representation of Phrases

A syntactic phrase can be represented in various ways. At the bottom end of the representation spectrum, a phrase can be represented simply by the unordered set of its words, disregarding *all* structure. At the other end, all linguistic structure can be taken into account, resulting in complicated parse-tree representations. The choice is a trade-off between syntactic information and the ease of phrase extraction.

For example, a simple noun phrase picker could easily be constructed by looking for sequences of articles, adjectives, and nouns within a text. A noun phrase extracted like that would contain little information about how its adjectives and nouns are related to each other, except that adjacent words are most probably more related than nonadjacent ones. In an unordered set-of-words representation, and assuming there is no special treatment of proper names, the noun phrase

the hillary clinton health care bill proposal

would contain *bill clinton*, but it is obvious that this phrase does not refer to him. However, experimentally such a co-occurrence of query keywords within a noun phrase has resulted in clear improvements in precision (Arampatzis et al., 1997a). A sequence-of-words representation does not contain *bill clinton* (rightly), but does not contain *clinton proposal* either (wrongly). A full linguistic parsing would result in a much more precise representation. The parse-tree would contain too much linguistic detail, however, most of which is unnecessary for indexing, as such details reflect mostly the syntactic description of the natural language used rather than the intended meaning. Since the goal is to derive adequately precise (for retrieval purposes) meaning from syntax, we will settle for less than full linguistic parsing. Linguistically motivated *light* parsing has already been shown to slightly improve retrieval results over the classic IR approximation to noun phrase recognition (Hull et al., 1996).

As a result, an intermediate representation of noun and verb phrases is desirable, eliminating structures that can be assumed not to be beneficial to IR:

Definition 3 (noun phrase for IR) *A core noun phrase NP, from an IR point of view, has the general form:*

$$NP\ = det^{*} pre^{*} head\ post^{*}\ ,$$

where

- *det (determiner) = article, quantor, number, etc.*
- *pre (premodifier) = adjective, noun, or coordinated phrase.*
- *head = usually a noun.*
- *post (postmodifier) = prepositional phrase, relative clause, etc.*
- *the asterisk (∗) denotes a list of zero or more elements.*

Pre- and postmodifiers may recursively include other NPs.

Definition 4 (verb phrase for IR) *A verb phrase VP, from an IR point of view, has the general form*[2]*:*

$$VP\ = subj\ kernel\ comp^{*}\ ,$$

where

- *subj (subject) = an NP (in the wide sense, including personal names and personal pronouns).*
- *kernel (verbal clause) = inflected form of some verb, possibly composed with other auxiliary verbforms and adverbs.*
- *comp (complements, such as object, indirect object, or preposition complement) = an NP or prepositional phrase (PP).*
- *the asterisk (∗) denotes a list of zero or more elements, depending on the transitivity of the verb (e.g., intransitive verbs have no complements, transitive verbs have an object, ditransitive have an object and indirect object).*

[2] This definition of verb phrase is non-standard, but the difference is not really important for IR. What we are talking about here is rather a (verbal) clause, i.e. a verb phrase (according to its standard linguistic definition) including its subject, object(s) and other complements.

In accordance with the above definitions, it is possible to perform a parsing arguably lighter than full linguistic parsing, while a reasonable amount of structural information will still be retained. An example parse-tree is given in Figure 6.1. This is rather compact in comparison with a full linguistic parsetree, which would easily have overrun this page for the same sentence. Of course it is important that the parser is able to deduce the correct (or at least the most probable) dependency structure in complicated phrases. As we will see next, some elements that are considered of little interest from an IR point of view (e.g., determiners, prepositions, auxiliaries, and adverbs), may be eliminated.

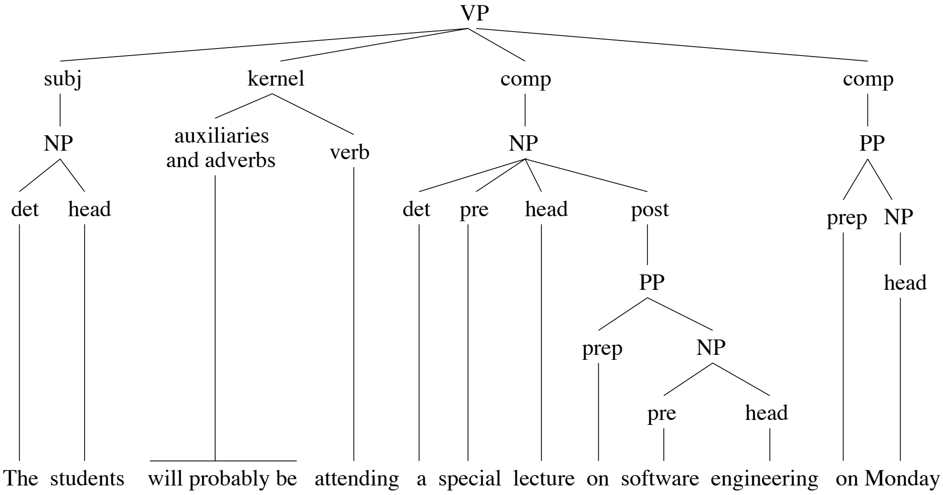


Figure 6.1: Light parsing for IR purposes.

6.5 Linguistic Normalization

The goal of normalization is to map different but semantically equivalent phrases onto one canonical representative phrase, the *phrase frame* (Figure 6.2). We distinguish between three types of normalization: the *morphological, syntactic*, and *lexicosemantic* normalization.

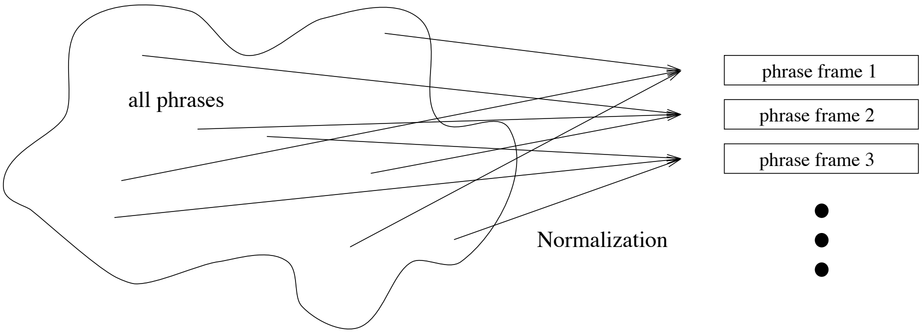


Figure 6.2: linguistic normalization

6.5.1 Morphological Normalization

Morphological normalization has traditionally been performed by means of stemming. Nonlinguistic stemming, especially when it operates in the absence of any lexicon at all, is rather aggressive and may result in improper conflations. For instance, a Porter-like stemmer without a lexicon will reduce both *university* and *universe* to *univers*, and *organization* to *organ*. Errors such as these are translated into a loss in retrieval precision. This impact is greater for more inflected languages than English because of the increased number of introduced ambiguities. Such improper conflations can be avoided by simply checking for the existence of the wordform in a lexicon after each reduction step. Nevertheless, the verb form *suited* will still be reduced wrongly to the noun *suite*.

Taking into account the linguistic context, a more conservative approach will prevent many of these errors. Conflations can be restricted to retain the part of speech of a word. In this respect, morphological normalization may be performed by means of *lemmatization*:

- Verb forms are reduced to the infinitive.
- Inflected forms of nouns are reduced to the nominative singular.
- Comparatives and superlatives of gradable adjectives are reduced to the absolute form.

For this task, the grammatical rules for forming, for example, past participles or noun plurals, should be applied in reverse. Furthermore, the utilization of exception lists in order to handle irregularities such as *wolf–wolves*, *bad–worse–worst*, and *see–saw–seen* is indispensable.

Lemmatization is relatively simple and handles mostly inflectional morphology. It is similar to the *lexicon-based word normalization*, as referred to in (Strzalkowski et al., 1999). It must be noted that there are cases in which lemmatization reduces noun and verb forms to the same lemma. Consider, for instance, the verb form *attacked* and the plural noun *attacks*; both will be lemmatized as *attack*. Although such conflations seem beneficial, there are empirical indications that the confusion between nouns and verbs when these are lemmatized, may slightly hurt effectiveness (Arampatzis et al., 2000d).

Derivational morphology involves semantics and cross part-of-speech word relations, and hence should be approached carefully. Certain derivational transformations may be suggested by syntax. For instance, verbs may be turned into nouns (nominalization) or the other way around, as will be shown in the next section. The remaining derivational morphology should be treated where possible by lexicosemantic normalization.

6.5.2 Syntactic Normalization

According to the linguistic principle of *headedness*, any phrase has a single head. This head is usually a noun (the last noun before the postmodifiers) in NP, and the main verb in the case of VP[3]. The rest of the phrase consists of modifiers. Consequently, every phrase can be mapped onto a *phrase frame*

$$PF = [h, m] \ .$$

The head h gives the central concept of the phrase and the list m of modifiers serves to make it more precise. Conversely, the head may be used as an abstraction of the phrase, losing precision but gaining recall. It should be noted that although the head–modifier relation implies semantic dependence, what we have here is purely a syntactic relation. The intention is to produce meaningful indexing terms without deep semantic analysis, therefore the precise semantic interpretation of any head–modifier relation is forborne, treating it simply as an ordered relation.

Heads and modifiers in the form of phrases are recursively defined as phrase frames: $[[h_1, m_1], [h_2, m_2]]$. The modifier part may be empty in the case of a bare head. This case is denoted equivalently by $[h,]$ or $[h]$. The head may serve as an index for a list of phrases with occurrence frequencies

```
[ engineering   1026 ,
      of software 7 ;
      reverse   102 ;
      software  842 ;
      ... ]
```

where the frequency of a bare head includes that of its modified occurrences. Alternative modifications of the head are separated by semicolons.

Phrases frames are produced by normalizing the phrase representations of Definitions 3 and 4. In noun phrases, determiners are of little interest for IR, thus they may be eliminated. The normalization of noun phrase is defined as

Definition 5 (noun phrase normalization)

$$NP \ = det^* pre^* head\ post^* \mapsto [head, pre^* post^*] \ .$$

The elements of the list *pre*post** are considered to modify the head *independently* of each other, and they are separated by semicolons, hence any PF containing a list, e.g., $[h, m] = [h, m_1; m_2]$, may be expanded as $[h, m_1]; [h, m_2]$. The noun phrase normalization can be applied recursively on heads and modifiers that include other NPs. For example

a special lecture on software engineering
↦ `[lecture, special; on software engineering]`
↦ `[lecture, special; on [engineering,software]]`.

[3] According to the standard definition of VP, the main verb is its head. As we will see later, the head of the (verbal) clause of Definition 4 is its subject.

Note that an alternative notation which would have eliminated the list of modifiers is

`[[lecture, special], on [engineering, software]]`.

However, such a notation rather implies that *special* is a more important modifier than `[engineering, software]`, but we prefer not to see it like this.

Prepositions (e.g., *on* in the last example) may optionally be kept for further semantic analysis, although their use is usually dropped for simplicity. It must be noted, however, that *the spaceman on the ship* enjoys a different view than *the spaceman outside the ship* and *the spaceman without ship* is probably not even in space. The impact of prepositions on retrieval performance is not well established, but their careful treatment may be beneficial. Their use and meaning can always be postponed until the matching of PFs. Prepositions, conjuctions and other such lexical items were considered as connectors in the characterization language of *index expressions* (Bruza and Weide, 1992).

The noun phrase presents only limited opportunities for syntactic normalization. For the verb phrase, more normalizations can be found that preserve its meaning (or rather do not lose information obviously relevant for retrieval purposes). To begin with the kernel, the elimination of time, modality, and voice seems reasonable. The obviously meaningful head–modifier combinations are [*subj*, *verb*] and [*verb*, *comp**].

Definition 6 (verb phrase normalization I)

$$VP \ = subj\ kernel\ comp^* \mapsto [subj, verb(kernel)]; [verb(kernel), comp^*]\ .$$

where verb(.) picks up the main verb of a kernel.

For example

the students will probably attend a special lecture on Monday
↦ `[the students, attend]; [attend, a special lecture; on Monday]`.

In Definition 6 the adverbs of the kernel are eliminated. Small experiments have suggested that adverbs have a little indexing value (Arampatzis et al., 2000d). They might be more useful, however, if they combine with the verbs (or adjectives in the case of noun phrase) they modify; for example, `[attend, probably]`. The indexing value of such verb–adverb and adjective–adverb pairs has to be evaluated empirically.

The possibility exists to map verbs to nouns (*nominalization*) or vice versa (*verbalization*). Such normalization allows the matching of PFs derived from different sources (verb phrases or noun phrases). For example, *(to) implement* can be nominalized to *implementation*. Since the opposite transformation is also possible for nominalized verb forms, the choice has to be made on the basis of experimentation. We will presently choose to turn everything into “pictures” (noun phrases) by applying the former alternative. This results in a more drastic (and compact) normalization:

Definition 7 (Verb Phrase Normalization II)

$$VP \ = subj\ kernel\ comp^* \mapsto [nom(verb(kernel)), subj\ comp^*]\ .$$

where verb(.) picks up the main verb from a kernel, and the function nom() nominalizes verbs.

For example

> *the students will probably attend a special lecture on Monday*
> ↦ `[attendance, the students; a special lecture; on Monday].`

Similarily, adverbs may be mapped onto adjectives to modify the nominalized verbs; for example, `[attendance, probable]`. Cross part-of-speech transformations such as those controlled by syntax can deal to some extent with derivational morphology, compensating for the conservative nature of lemmatization described in the previous section. The further application of the noun phrase normalization to the last phrase frame eventually results in

> `[attendance, student; [lecture, special]; on [Monday]].`

All these normalizations are rather language-dependent, and the final decision of what has to be included in the phrase frames should be left to the linguists and system designers; we have merely suggested a few obvious ones.

6.5.3 Lexicosemantic Normalization

This kind of normalization depends on the observation that certain relations can be found between the meaning of individual words. The most well known of those lexicosemantic relations are

- *synonymy* and *antonymy*,
- *hyponymy* and *hypernymy* (the *is-a* relation),
- *meronymy* and *holonymy* (the *part-of* relation).

Two important aspects that should be taken into account for this kind of normalization are *polysemy* and *collocations*.

A word is polysemous if its meaning depends on the context. For example, by itself the noun *note* can be meant as a being a short letter, or as a musical note; consequently its context has to clarify its meaning. The intended meaning determines the words that are lexicosemantically related to the initial word. Using the synonymy relation for the first meaning we can obtain *brief*, while *tune* is obtained in the second case. This suggests that the conceptual context of a word should be taken into account.

Collocations are two or more words that often co-occur adjacent to one another (e.g., *health care*) having in this combination a certain meaning. When using WordNet in expanding a query with hypernyms, the notion *health care* obtains *social insurance*, which cannot be obtained in any case by expanding the two separate words. This observation suggests that collocations should be considered as single units.

Assuming that the word sense ambiguity originating from polysemy is resolved, three possibilities can been seen for lexicosemantic normalization.

1. **semantic clustering** in analogy with stemming. For instance, several synonyms in a context are reduced to one *word cluster*. The word cluster may be represented by the most frequent of the synonyms.

2. **semantic expansion**, expanding a term with all its -nyms. The derived terms may be weighted according to their relation with the initial term.

3. **incorporation of a semantic similarity function into the retrieval function (fuzzy matching)**. Based on a semantic *taxonomy*, an *ontology*, or a *semantic network* we can define a *semantic similarity function* for words.

Semantic clustering is rather aggressive and suffers from the same drawbacks as stemming. For example, two "synonyms" are often merely overlapping in meaning and they do not actually mean the same thing. The choice to call them synonyms depends on the degree of overlap. One of the questions is how extended these clusters should be; that is, what maximum semantic distance is allowed for two words in order for them to belong to the same cluster. Again, usability and discrimination come to play an important role here. Clusters that are too large will be assigned as indexing terms to too many documents and therefore are not discriminating. Clusters that are too small (e.g., one or two synonyms) will not have a great impact in performance compared to conventional indexing; thus they are not usable. Experimentation should provide a usably discriminating cluster size. Semantic expansion can partly overcome the cluster size problem by supplying many related terms weighted inversely proportional to their semantic distance from the original term. Expansion can easily result in an explosion of indexing or query terms, however. The possibility of fuzzy matching seems elegant and exciting, although it is far more computationally expensive than the others.

Working out fuzzy matching a bit more, using only the relations *SYN*onymy, *HYPON*nymy, and *HYPER*nymy between two words x and y, one could define

$$sim(x,y) = \begin{cases} 1 & x = y \\ 0.9 & x \in SYN(y) \\ 0.7^n & x \in HYPON_n(y) \\ 0.5^n & x \in HYPER_n(y) \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

where $a \in HYPER_n(b)$ means that a can be found by walking in the graph of hypernyms of b a number of n steps; $a \in HYPON_n(b)$ is similarly defined. *SYN* is a symmetric relation, meaning that if $x \in SYN(y)$ then $y \in SYN(x)$, so it is sufficient to check only if one of the two holds. It should be noted that *sim* assumes an order in its arguments, x is a word from a document, and y is from a query. Moreover, hypernyms of query terms are matched with lower weights than hyponyms to reflect the assumption that a user's query *salmon* should not retrieve many documents about *fish* in general, but *fish* should retrieve documents about *salmon*.

As an example of fuzzy matching, consider the sentence

the students will probably attend a conference on software engineering,

from which, after syntactic and morphological normalization and the elimination of some (assumed) redundant elements, the following phrase frame may be constructed:

```
[attendance, student; conference; [engineering, software]].
```

Now let us consider another sentence

> *the pupils are listening carefully to the tutorial about software engineering,*

which in a phrase frame representation becomes

> `[listening, pupil; tutorial; [engineering, software]]`.

Note that `listening` here represents the nominalized form (*the listening*) of the verb *to listen* rather than its progressive form. Using WordNet's lexical graph, and assuming that the latter sentence is part of a natural language description of a user's information need (query), the following relations hold

$$\texttt{student} = \mathit{SYN}(\texttt{pupil}) \Rightarrow \mathit{sim}(\texttt{student}, \texttt{pupil}) = 0.9 \ ,$$

$$\texttt{conference} = \mathit{HYPER}_2(\texttt{tutorial}) \Rightarrow \mathit{sim}(\texttt{tutorial}, \texttt{conference}) = 0.5^2 \ .$$

The nouns *listening* and *attendance* may be matched through the relation that holds between their corresponding verbs.

$$\texttt{attend} = \mathit{HYPON}_1(\texttt{listen}) \Rightarrow \mathit{sim}(\texttt{attend}, \texttt{listen}) = 0.7 \ .$$

Using these relations, it is now easy to match the two sentences. However, this example is conveniently selected as it results in phrase frames with similar structures. In general, this is not the case, suggesting that such a lexicosemantic similarity function should be a part of a larger structural matching technique.

6.6 Weighting and Matching

Term weighting is a crucial part of any IR system. Statistical weighting schemes such as *tf.idf*, which perform well for single terms, do not seem to extend on multiword terms. Most work on the use of multiword indexing terms in IR has concentrated on representation and matching strategies. Little consideration was given to phrase weighting and to scoring of documents matched. An obvious weighting strategy for phrasal terms is to weight a term as a function of the weights of its components. However, such strategies have not produced uniform results (Fagan, 1987; Lewis and Croft, 1990). We suggest a simple weighting scheme suitable for phrase frames that takes into account the modification structure and its depth.

Phrase frames may contain nested phrase frames (subframes) at different depths. To simplify the structural matching of complicated phrase frames, the strategy of *unnesting* (Koster et al., 1999) can be followed. The unnesting of a phrase frame produces all possible subframes down to single-term frames. This can be understood more easily by visualizing a phrase frame as a tree; the root node is the main head, and every node is modified by its child nodes. Such an abstract tree is depicted in Figure 6.3. Unnesting produces all possible triangles q of all possible sizes and depths. The main head of a frame carries the most semantic information of all the other elements in the frame. The other elements modify the head, increasing the amount of semantic information carried

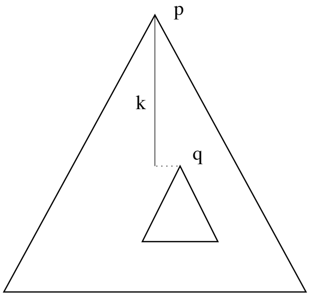


Figure 6.3: Tree visualization of a phrase frame p with a subframe q at depth k.

by the frame. The amount of information added to the frame by an element decreases with the depth of the element within the frame[4].

First we introduce the predicate $sub(p,q,k)$ as a shorthand for the expression: phrase frame p has phrase frame q as a subframe at depth k. The *depth weight* dw of subframe q obtained from frame p can be expressed as

$$dw(q,p) = \sum_{k:sub(p,q,k)} f(k) \ , \tag{6.2}$$

where f is some decreasing function, e.g. $f(k) = \frac{1}{1+k}$. The sum accounts for more than one occurrence of subframe q within p (rather rare because stylistic considerations for natural language do not favor repetitions of the same subphrase within a NP or VP). Let document d have the set $C(d)$ of phrase frames as characterization, augmented with all the unnested terms down to single terms. Then the *frame frequency* of q within document d can be described as

$$f\!f(q,d) = \sum_{p \in C(d)} dw(q,p) \ . \tag{6.3}$$

The *geometrical length* of the document frame vector in the frame space is

$$l(d) = \sqrt{\sum_{q \in C(d)} f\!f(q,d)^2} \ . \tag{6.4}$$

The *weight* of frame q within document d is estimated by

$$w(q,d) = \frac{f\!f(q,d)}{l(d)} \ . \tag{6.5}$$

The similarity between document d and query q then can be estimated by the dot product formula

$$S(d,q) = \sum_{r \in C(d) \cap C(q)} w(r,d) * w(r,q) \ . \tag{6.6}$$

[4] This is rather a questionable assumption. One can find examples where the information added to the frame decreases with the depth of the element, while other examples may support that such a use of an element's depth may be misleading. Whether or not taking the depth into account is beneficial for retrieval effectiveness should rather be determined experimentally.

Using the last formula, the documents of a collection can be ranked in a response to a query. Of course, Equation 6.6 may be modified to incorporate some form of IDF. Without an empirical evaluation of this scheme, however, not much more can be said.

6.7 An Architecture of an LMI System

In this section, we discuss how a retrieval system based on the linguistically motivated indexing (LMI) model described in this chapter can be implemented. Until now we assumed that an immaculate linguistic analysis is available, disregarding technical implementation details. However, trying to put such a retrieval system into practice, the inefficiencies and ineffectiveness of currently available NLP techniques become apparent. A major source of ineffectiveness is *linguistic ambiguity*, some of which can be resolved, while the rest requires sophisticated semantic analysis. Furthermore, NLP can be so time-consuming that it becomes impractical for real-world applications. Lacking deep semantic analysis, some design decisions have to be made in order to make a linguistically motivated retrieval system usable in the real world.

Given a collection of text documents, the indexing task assigns to each document a characterization in the form of (weighted) phrase frames. Phrase frames are derived from documents through a sequence of processing steps.

1. Tokenization.
2. Part-of-speech tagging.
3. Morphological normalization.
4. Collocation identification.
5. Lexicosemantic normalization.
6. Syntactic analysis.
7. Syntactic normalization.
8. Weighting.

The tokenization step constitutes the detection of sentence boundaries followed by the division of sentences into words. This may sufficiently be implemented based on capitalization rules, spacing, tabbing, and document layout considerations.

Part-of-speech tagging assigns a part-of-speech label to each word in a text, depending on the labels assigned to the words around it. It is possible that more than one label can be assigned to a word, suggesting some kind of *lexical ambiguity* in the input. A simple way to overcome this ambiguity is to retain only the most probable label for an ambiguous word, based on the occurrence frequencies of the word under all its possible parts of speech. Another solution would be to postpone lexical ambiguity resolution until syntactic analysis. Syntactic rules are able to resolve some lexical ambiguity, but not all. Taking collocations as single units may also resolve some lexical ambiguity. For example, while *social* can be either an adjective or a noun, *social security* taken as a single unit is a noun collocation because it functions as a noun. After part-of-speech tagging, morphological normalization is performed, guided by the assigned labels.

Static collocation lists or word co-occurrence statistics can be used to identify collocations. Identified collocations are treated as single units in subsequent processing steps. Lexicosemantic normalization is the following step, assuming that it is implemented by semantic clustering or expansion. If it is implemented as a semantic similarity function, then it is performed during the matching of documents to queries rather than during indexing.

Syntactic analysis or parsing reveals syntactic relations between words, collocations, and phrases in a sentence. Syntactic relations are identified based on syntactic rules (grammar). Given the part-of-speech information for a text, syntactic rules can be formulated for sequences of part-of-speech labels; for example, the combination adjective–noun surrounded by other part-of-speech labels is a noun phrase. Structural ambiguity — what modifies what — may occur during analysis. For instance, every noun phrase with three or more words, two or more of which are nouns, is a potential source of structural ambiguity. To disambiguate such structures, statistical methods can be applied. In the case of noun phrases, first, frequency information is collected from the corpus for all two-word noun phrases. Then all three-word noun phrases are disambiguated by assigning to them the most probable structure based on the frequencies of two-word noun phrases. Gradually this can be applied up to n-word noun phrases based on the frequencies of all previously disambiguated k-word noun phrases ($k < n$). *Left-dependence* may be assigned where not enough frequency information is available, since it is the most probable modification structure in the English noun phrase. A similar statistical approach can be developed to resolve the *prepositional phrase attachment* problem, guided by subcategorization information about nouns and verbs.

The next step, syntactic normalization, may be incorporated in the parser in a way that the parser outputs regularized parse tree representations (e.g., phrase frames). As soon as the collection of documents is translated into a phrase frame representation, phrase frames can be weighted according to their frequency characteristics and structure.

A similar procedure to the above indexing steps can be followed to turn a natural language query into a phrase frame representation, allowing the matching of queries to documents. The indexing procedure just described can replace the indexing part of a conventional retrieval system architecture. There is no obvious need why radical architectural changes should be made. Inverted files, vector space, and probabilistic retrieval models are still suitable and may be adapted to work with linguistically-motivated indexing terms. What really changes is the way that indexing terms are extracted from documents and how these are matched. The current inefficiencies and ineffectiveness of NLP techniques can be treated for the time being by such statistical solutions as the (crude) ones described above. Fortunately, the explosion in computational power that becomes available daily, combined with the efforts put into NLP issues from the (computational) linguists' side, suggests that the use of linguistically motivated retrieval systems in everyday practice is merely a matter of time.

6.8 Summary

The bag-of-words paradigm has dominated commercially available information retrieval systems for about three decades. The main reasons for the endurance of such systems based on such simple assumptions as the naive keyword retrieval hypothesis are first, that they are relatively easy and simple to implement (it takes a third-year computer science student with the knowledge of a programming language, an IR textbook, and some days' time), and second and most important, that these systems have presented until recently a satisfactory effectiveness in searching collections in the class of megabytes.

The digital and networking revolution has made available data in the class of gigabytes, exposing the inadequate nature of keyword-based systems. The searching for information has become a laborious task for a user who presently has to perform her or his own selection over the "dirty" and very lengthy output of a World Wide Web search engine, for example. As a consequence, many researchers have aimed at higher levels of natural language utilization in IR, assuming that better "understanding" of the information need as well as the information residing in a database is the key for improving retrieval effectiveness.

The attempts made to break out of the bag-of-words paradigm by employing NLP and other linguistic resources have until now presented inconsistent or at least dubious results, however. One explanation of why NLP has not had more successes in IR is that it does not go far enough. First, the currently available NLP techniques suffer from lack of accuracy and efficiency, and second, there are doubts if syntactic structure is a good substitute for semantic content. The evidence so far suggests further investigation and better modeling.

In this chapter, we have reviewed some of the most important research in the field, and discussed a general model for a linguistically motivated retrieval system. We believe that a retrieval schema based on the phrase retrieval hypothesis and incorporating linguistic normalization has more potential in improving retrieval effectiveness than keyword-based schemas. We have suggested a suitable model and some techniques, however, whether or not the discussed techniques work is still not entirely clear. We will present our empirical evaluation of some parts of the model in Chapter 7.

Considering that better IR means more user satisfaction, perhaps a more radical change in the focus of IR is needed. Maybe the future of IR is not to provide better ranking of retrieved documents but to supply the very information a user is seeking. A compact summary of retrieval results or a brief answer might be more usable for an average user than a ranked list of hundreds of documents. Automatic summarization, question answering, and information extraction systems require advanced NLP techniques, however. Furthermore, the traditional precision- and recall-based retrieval quality metrics may not be able to evaluate the ability of a system to derive such information; consequently other metrics will have to be developed. Nevertheless, one thing seems certain for the future: NLP and other linguistic resources will become — if they are not already becoming — indispensable parts of every effective IR system.

Chapter 7

An Evaluation of LMI Schemes

This chapter describes our experimental work in using *linguistically motivated indexing (LMI)* schemes for information seeking tasks. It is based on our previously published work in (Arampatzis et al., 1997a) and (Arampatzis et al., 2000d).

We describe two experiments. The first is a small retrieval experiment which was performed with the IRENA system in 1995. Its results led us to the development of the linguistically-motivated indexing model which we introduced in Chapter 6. The second experiment sets out to evaluate a part of this model in a document classification context. We will see the implications that both experiments have for text filtering environments.

7.1 The IRENA System

The experimental IRENA (Information Retrieval Engine based on Natural language Analysis) system was developed in 1995 in order to study the impact of natural language processing (NLP) techniques on the precision and recall of document retrieval systems. The NLP component deals with the morphological and lexicosemantical part of the English language to improve recall, and with syntax to improve precision.

IRENA accepts queries in the form of noun phrases. For the syntactical analysis of queries, the power of the AGFL[1] formalism was explored in describing and developing a syntactical analyzer for the English noun phrase. The parser syntactically analyzes NP queries to extract only adjectives and nouns, or any other words which function as such, e.g., proper names, gerunds, or adjectival present and past participles.

Morphology and lexicosemantics are dealt with by *query expansion*. Keywords can be lexicosemantically expanded with synonyms. Moreover, keywords can also be expanded with morphological variants of them, for instance, plurals for nouns or comparative and superlative forms for adjectives.

The system was tested on a small corpus of manually collected documents about pop music. The retrieval strategy was based either on noun phrases or on proximity considerations. Next, we will briefly describe how lexicosemantical and morphological expansion was performed, and the approach we followed in using noun phrases. We will report the results of a small experiment, and reinterpret these in the contexts of the linguistically

[1] Affix Grammars over Finite Lattices; see `http://www.cs.kun.nl/agfl/`

motivated indexing model of Chapter 6 and relevance feedback environments such as filtering. We will not expand on how syntactic analysis was performed; the interested reader should refer to (Arampatzis et al., 1997a) or (Arampatzis and Tsoris, 1996).

7.1.1 Lexicosemantic Query Expansion

Synonyms of the keywords are obtained from WordNet (Miller, 1995). In WordNet, nouns, verbs, adjectives and adverbs are organized into synonym sets (*synsets*), i.e., lists of synonymous wordforms that are interchangeable in some context.

If we disregard word meaning and combine all synsets that a keyword belongs to, each keyword of the query "popular bands" will be expanded as follows:

popular, `demotic, lay, plain, nontechnical, unspecialized, untechnical, pop.`

band, `set, circle, lot, stria, striation, banding, stripe, dance band, dance orchestra, frequency band, ring.`

Words like "demotic" and "stria" are not synonyms of "popular" and "band" in music contexts. To avoid such erroneous expansions, Irena presents every candidate synset to the user, asking for a confirmation of its relevance before using it for expansion. As a result, for the keyword "band" of the query considered, a user should reject the synsets

{ band, stria, striation }
`(a stripe of contrasting color;`
`‘‘chromosomes exhibit characteristic bands’’)`

{ band, frequency band }
`(band of radio frequencies for e.g. transmitting a TV signal)`

and select only the following:

{ dance band, band, dance orchestra }
`(a group of musicians playing popular music for dancing)`

In this way, the ideal expansion of the query has as follows:

popular, `pop.`

band, `dance band, dance orchestra.`

7.1.2 Morphological Query Expansion

The system supports (inflectional only) morphological expansion of keywords. Expansion can be applied only to nouns and gradable adjectives. Adjectival participles and gerunds are not expanded at all, due to the fact that adjectival participles are not inflected and gerunds have no plural.

The expander is capable of conflating, following the English grammar rules (Alexander, 1988), the singular or plural forms (depending on which form is missing from the query) of nouns, and the nominative, comparative, or superlative forms of gradable adjectives. A lists of nouns with irregular plurals, and a list of gradable adjectives are used for these purposes.

All common nouns in English fall into one of two sub-classes: they may be either *countable* or *uncountable* and that distinction is fundamental for the existence of the plural. Unfortunately, strict classifications of nouns are in many cases unreliable, as some nouns which are normally uncountable can be used as countable in certain contexts. For instance, the noun “weather” is normally uncountable, but one can say “I go out all weathers”. The distinction of nouns in countable and uncountable is not taken into consideration, so some nouns may be expanded into non-existent plural forms.

7.1.3 The Noun Phrase as a Unit of Co-occurrence

An ideal retrieval strategy based on noun phrases would require some measure of the “nearness” of one noun phrase (in the query) to another (in the document). Although similar measures have been developed, e.g., *logical nearness* in (Bruza, 1993; Bruza and IJdens, 1994), we investigated in IRENA other more heuristic strategies, namely, the *noun phrase co-occurrence hypothesis*.

Our basic premise was that words occurring in the same noun phrase share some semantical relation. If two or more nouns or adjectives co-occurred in a single noun phrase, then we assumed that they share some relatedness, even without knowing what they stand for. For example in the phrase

> *. . . tracks were recorded at the BBC studios for later radio programs.*

the nouns “radio”, “programs” and the proper name “BBC” which reside in the same (underlined) NP[2] are semantically related. If a retrieval strategy requires all keywords of the query “radio programs on BBC” to co-occur in the same NP, then the above document will be retrieved, while the following two will not:

> Document 1: *The transmission of his first* **radio programs** *resembled the early years of the creation of* **BBC** *empire which . . .*
>
> Document 2: *Ten musicians from the* **BBC** *Symphony Orchestra were interviewed in several* **radio programs** *of L.A. stations . . .*

[2] Syntactically, the prepositional phrase (PP) “for later radio programs” belongs to the verb phrase in this example. The problem here is that a simple-minded parser cannot resolve this PP-attachment. We used an NP-parser only, which was not able to parse verb phrases. In this respect, we went for the *longest possible* NP resolving structural ambiguity by means of *syntactic under-specification*.

In both of the documents above, the three words of the query do not all reside in the same NP. These real-world documents clearly show that extra linguistic processing is superior compared to a *proximity search* that requires words of the user's query to be just close to each other in a document.

Of course, there are exceptions which do not conform to the NP co-occurrence hypothesis. As an example, let us consider the following query and document:

> Query: *soundtracks of films*
>
> Document: *In this album, there is a good background, but there is something missing. Either a solo voice or instrument. Or at least a film. Soundtrack without pictures so to speak.*

The noun "picture" is a synonym of the noun "film" and belongs to the same NP as "soundtrack". The meaning of the last sentence is merely that this album could be a soundtrack of a movie, but it was actually not. Note that prepositions (e.g., "of' or "without") are not considered at all by the NP co-occurrence hypothesis.

7.1.4 An Experiment

We conducted a small experiment using a manually collected corpus of documents about pop music (e.g. magazine articles, FAQs about artists, interviews, reviews, etc.). Some corpus statistics are shown in Table 7.1.

Number of documents	633
Mean words per document	1695
Number of words in collection	1,072,762
Total size	6,752 Kb

Table 7.1: Statistics of the corpus used in the Irena experiment.

Two computer science students formulated 44 NP queries in total. The queries were parsed and an average of 2.6 keywords per query were extracted. The expansion with synonyms resulted in 4.1 additional search terms, that is about 1.6 synonyms per initial keyword. The morphological expansion of all keywords and their synonyms added an average of 14.2 search terms. Consequently, the fully expanded queries were 20.9 terms on average.

All search terms were unweighted; the searches were boolean. Three different kinds of searches were compared:

K: all initial keywords ANDed.

KM: each initial keyword was ORed with all its morphological variants. Then, all batches of ORed search terms were ANDed.

KSM: each initial keyword was ORed with all its synonyms and all their morphological variants. Then, all batches of ORed search terms were ANDed.

The retrieved sets were restricted to the documents in which the above search types were satisfied within some text window. Therefore, the rankings were created by varying the text window: documents which satisfied the search within an NP were ranked at the top, followed by the documents with term co-occurrence within 2 text lines, then 3 text lines, and so on. Test runs showed that using this kind of ranking, precision decreased and recall increased by going down rank positions.

The precision–recall results are summarized in Table 7.2. The retrieval results per query were judged for relevance by the user who submitted the query. We were confronted with the classic problem of calculating recall, so we report the *relative recall* instead: the configuration which retrieved the most relevant documents for a query was assumed to had achieved 100% recall. Precision and recall were macro-averaged across all queries.

	K		**KM**		**KSM**	
Window	Prec. %	R-Recall %	Prec. %	R-Recall %	Prec. %	R-Recall %
NP	100.00	6.31	95.65	23.14	91.38	27.90
2	79.17	19.98	76.45	49.44	71.61	58.41
3	74.60	24.75	70.62	58.41	66.33	69.43
4	72.97	28.40	68.05	66.83	63.40	78.40
6	70.21	34.71	65.22	76.79	59.59	91.58
8	69.44	39.48	63.87	84.22	56.38	100.00

Table 7.2: Precision-Recall results.

The NP co-occurrence requirement resulted in high precision levels, above 90% for all search types. However, the recall was extremely low compared to simple co-occurrence in a text window. As the window size increases from 2 to 8 lines, recall is gained at the price of a drop in precision. It is found, however, that by increasing the window size to more than 16 lines, precision is dramatically lowered to 25-35%. Upon enlarging the window, keywords may appear in different paragraphs with possibly different subjects. A window size of 4 to 8 lines gives reasonable levels of precision and recall.

Expanding queries with synonyms and morphological variants led to a marked increase in recall. The drop in precision can be considered as insubstantial compared to the recall gained.

7.1.5 Discussion

With current standards, the IRENA experiment is an exercise rather than a full-scale experiment. The collection was small, but the techniques applied — e.g. natural language parsing, query expansion with synonyms, (manual) word-sense disambiguation — are still not so common in everyday information retrieval engines. IRENA has been a demonstration of the feasibility of the techniques.

The small-scale experiment has demonstrated that lexical and morphological expansion (normalization) of queries is indispensable for high recall and results in an insubstantial average loss of precision, hence, it is highly recommended. The NP co-occurrence criterion seems to be successful in determining whether keywords are semantically related, and achieves a much better precision than *proximity search*. The low recall obtained suggests the generalization of the NP co-occurrence hypothesis to wider classes of phrases to delimit the semantic relatedness between words, e.g. verb phrases.

The low recall achieved could be interpreted in two different ways. One could argue that use of the noun phrase shows no promise in improving the performance of IR systems (Smeaton, 1997; Gay and Croft, 1990). We argue, on the other hand, that we should retain the noun phrase as a unit of co-occurrence, but should investigate the possibilities of enhancing the recall without losing too much precision, by taking into account linguistic variation and anaphora.

What are the implications of the experiment for relevance feedback environments? Let us consider an adaptive filtering task. A user issues a short query, and there is no other relevance information. An adaptive system counts on user relevance judgments for documents retrieved, so as to adapt the filtering model and improve the query. However, explicit user feedback is not guaranteed in a real-world system, so the system should be able to infer relevance from other factors, such as the time the user has spent in reading a document. The NP co-occurrence criterion can provide an additional means for inference. If co-occurrence of the search-words in the same noun phrase is strong evidence for relevance, then explicit user feedback becomes less important for those documents. In other words, we suggest a *rule-based* pseudo-relevance (blind) feedback: if some kind of NP co-occurrence holds in a retrieved document, then use the document as a relevant training example.

7.2 An Evaluation of some LMI Schemes

In this section, we describe an experiment which was performed in the context of validating the LMI scheme described in Chapter 6. The approach taken is based on a part-of-speech (POS) tagger and syntactic pattern matching.

First, we experimented with representations based on combinations of different POS categories. These representations combine the category of nouns with that of adjectives, verbs, and adverbs. The different representational choices are compared to the baseline of using all keywords as index terms. Then, we experimented with composite terms which were built, firstly, using a simple criterion like *word adjacency*, and secondly, using syntactic structure like *word modification*. We also investigated the effect of morphological normalization by means of *lemmatization*, which can be seen as POS-directed stemming. Evaluation is done in a document classification environment using 11-point recall-precision, and average interpolated precision.

What is new in this approach is the variety of schemes evaluated. It is important since it should not only help to overcome the well-known problems of bag-of-words representations, but also the difficulties raised by non-linguistic text simplification techniques such as stemming, stop-word deletion, and term selection. Our findings apply to information retrieval and most of its related areas.

7.2.1 Unnesting to Binary Terms

Our long-term goal is to validate empirically the LMI model we introduced in Chapter 6. We do not yet have a suitable way of structurally matching phrase frames.

In order to simplify the structural matching of phrases, and also to raise recall, we currently follow the strategy of *unnesting* all complicated phrase frames (Koster et al., 1999). A composed term like $[a, [b, c]]$ is decomposed into two frames $[b, c]$ and $[a, b]$ using b as an abstraction for $[b, c]$. When this decomposition is applied recursively, it results in *binary terms* (BTs). As an example, consider the sentences

A student visits a conference on software engineering.
The software engineering conference is visited by some students.

from which, due to syntactical and morphological normalization, the same two frames are initially constructed for both sentences:

$$BT_1 = [\texttt{student}, \texttt{visit}],\ PF_1 = [\texttt{visit}, [\texttt{conference}, [\texttt{engineering}, \texttt{software}]]].$$

PF_1 is further unnested to

$$BT_2 = [\texttt{visit}, \texttt{conference}],\ BT_3 = [\texttt{conference}, \texttt{engineering}]\ ,$$

$$\text{and}\quad BT_4 = [\texttt{engineering}, \texttt{software}]\ .$$

Of course the unnesting makes it all the more important that a syntactical analyzer should be able to deduce the right dependency structure in complicated phrases.

In the current phase of our experimentation, phrase frames are constructed only from noun phrases, taking into account only prepositional phrase (PP) post-modifiers of nouns starting with the preposition "of". These PPs are more likely to modify the preceding noun than others for which the PP-attachment problem has to be solved. However, we were able to disambiguate the modification structure of complicated noun phrases by applying statistical methods (see Section 7.2.3.1). We did not apply any lexico-semantical normalization.

7.2.2 Representational Choices

The different indexing sets we experimented with are summarized below. The acronyms will be used to refer to these choices in the rest of this chapter.

w (**w**ords): All word-forms found in text.

Sw (**S**temmed **w**ords): All word-forms stemmed by a Porter stemmer. This is a traditional indexing scheme and serves as the baseline in order to compare the effectiveness of the rest of the indexing schemes.

Lw (**L**emmatized **w**ords): The same as **w**, except that all word-forms are lemmatized with respect to their POS category. In all the following choices, lemmatization is applied as standard.

Of course, for all **w, Sw** and **Lw** we eliminate words of low indexing value by using a *POS stop-list* (Section 7.2.3.1).

Ln (**L**emmatized **n**ouns): Nouns and proper nouns are well-known to be important in retrieval. What happens if we omit all other keywords?

Lnj (**L**emmatized **n**ouns and ad**j**ectives): The combined effect of using the union of nouns and adjectives is investigated in this experiment. These two categories cover most of the words occurring in noun phrases.

Lnv (**L**emmatized **n**ouns and **v**erbs): We investigate the combined effect of using the union of nouns and verbs.

Lnjv (**L**emmatized **n**ouns, ad**j**ectives and **v**erbs): This experiment serves as an indication of what might happen if we include to the indexing language only linguistic entities which are extracted from noun or verb phrases. Moreover, the impact of using adverbs for indexing can be measured indirectly by comparing **Lnjv** to **Lw**, since the indexing set **Lnjv** can be constructed from **Lw** by removing adverbs.

Lap (**L**emmatized **a**djacent word-**p**airs, extracted from NPs): These word-pairs consist of the nouns and adjectives of **Lnj**, associated to form 2-word phrases by using the adjacency criterion. The hypothesis for this experiment is that adjacent words can be considered semantically related because of their proximity and be taken as one term. We use an extended notion of adjacency by accepting non-adjacent words as adjacent if the in-between words belong to certain POS categories. In fact, these were all POS categories except the categories of nouns and adjectives (e.g., determiners, articles, or prepositions). For instance, the phrase "pollution of the air" gives the adjacent pair `pollution_air`. The initial word order was retained.

This is an important experiment because its comparison to **Lbt** (described next) should measure the effect of syntactical normalization.

Lbt (**L**emmatized **b**inary **t**erms (**Lbt**, extracted from NPs): These binary terms consist of the nouns and adjectives of **Lnj**, associated to form 2-word phrases by using the term modification criterion, i.e. head–modifier pairs. The head–modifier pairs are computationally more expensive than adjacent pairs since syntactical normalization is required, however, binary terms are syntactically canonical, e.g. both phrases "air pollution" and "pollution of the air" are mapped onto the same head–modifier pair, `[pollution,air]`.

7.2.3 Experimental Setup

Our main concern is to evaluate different indexing schemes. Document classification, categorization, or routing environments provide a good test-bed for such evaluations. In such environments, given a pre-classified corpus, the measurement of recall is straightforward, while for classical retrieval systems it involves expensive human judgments.

The experimental system is based on the vector space model with a dot-product similarity function (Section A.1), terms are weighted in a *ltc* fashion (Section A.2.1), and classifiers are constructed automatically using Rocchio's relevance feedback method (Section A.4). We used the original Rocchio formula, that is, $\alpha = 0$ and $\beta = \gamma$.

Instead of making a binary decision to either assign a document to a class or not, we allow the system to return a traditional ranked list of documents for every class: most relevant first, least relevant last. Thus, evaluation is done with 11-point interpolated recall–precision and average precision (Section A.5.2) on the Reuters-21578 text categorization test collection. (Section A.6.1).

Since there is a large variation in the numbers of relevant training documents per topic in the Reuters collection, we evaluate separately on small and large topics. As small topics are considered the ones which have ten or less training documents (32 topics in total), and the rest (58 in total) are considered as large.

7.2.3.1 Collection pre-processing

In order to obtain the appropriate indexing terms from the dataset for every experiment, we applied some pre-processing. The pre-processing was performed in six steps:

1. **Tokenization** (script written in PERL): Detection of sentence boundaries followed by division of sentences into words.

2. **Part of speech tagging**: Brill's rule-based tagger[3] (Brill, 1994) was employed to obtain POS information for the contents of the dataset. The tagger comes with a lexicon derived from both the Penn Treebank tagging of the Wall Street Journal (WSJ), and the Brown Corpus. Conveniently, the WSJ articles are, like the Reuters documents, about economic topics, thus this increased the reliability in tagging the Reuters corpus.

3. **Shallow parsing and term extraction** (script written in PERL): Syntactic pattern matching on the POS tags to extract noun phrases for the **Lap** and **Lbt** experiments. For the **Lbt** experiment, the extracted noun phrases were further syntactically normalized and unnested, while for the **Lap** they were just broken down to adjacent word-pairs. For the rest of the experiments, the corresponding terms were extracted based on the POS tags.

4. **POS stop-listing** (only for **w, Sw**, and **Lw**): It is well-known that the use of a stop-list improves the quality of an indexing set. Traditionally, a stop-list is constructed by taking a predetermined list of common function words. Since our approach is based on a POS tagger, we used a *POS stop-list* to remove all words belonging to the following categories: determiners (e.g., "a", "the", "all"), prepositions and subordinating conjunctions (e.g., "in", "to", "of"), coordinating conjunctions (e.g., "and"), pronouns (e.g., "I", "yours"), the infinite marker "to", modal verbs (e.g., "would", "must"), and all forms of the verbs "to be" and "to have".

5. **Disambiguation of the NP structure** (only for **Lbt**, PERL script): Noun phrases with more than two words can be structurally ambiguous. To disambiguate the modification structure we applied statistical methods. First we collected frequency information from the corpus for all noun phrases with two words. Then

[3] Eric Brill's tagger V1.14 and a description are available by anonymous ftp from: `ftp://ftp.cs.jhu.edu/pub/brill` in the `Programs` and `Papers` directories.

all 3-word noun phrases were disambiguated by assigning to them the most probable structure based on the frequencies of 2-word noun phrases. Gradually, this was applied up to n-word noun phrases based on the frequencies of all previously disambiguated k-word noun phrases ($k < n$). Where not enough frequency information was available, *left-dependence* was assigned since it is the most probable modification structure in the English noun phrase.

6. **Morphological Normalization**: Lemmatization was performed according to the POS information by using WordNet's v1.6 (Miller, 1995) morphology library functions[4].

   For **Sw**, words were stemmed using the Porter stemmer of the `Lingua::Stem` (version 0.30) extension to Perl.

7.2.4 Results and Discussion

Table 7.3 summarizes the average interpolated precision results of all experiments and their percentage change with respect to the baseline of **Sw**, the traditional indexing approach.

	small topics		large topics	
run	av. prec.	change	av. prec.	change
w	0.525	−2.2%	0.696	+0.4%
Sw	0.537	baseline	0.693	baseline
Lw	0.547	+1.9%	0.693	0.0%
Ln	0.559	+4.1%	0.678	−2.2%
Lnj	0.563	+4.8%	0.695	+0.3%
Lnv	0.540	+0.5%	0.683	−1.4%
Lnjv	0.548	+2.0%	0.694	+0.1%
Lnj+Lap	0.633	+17.9%	0.730	+5.3%
Lnj+Lbt	0.620	+15.4%	0.732	+5.6%

Table 7.3: Average precision results.

7.2.4.1 Stemming vs Lemmatization

The experiments with unstemmed, stemmed and lemmatized words (**w, Sw** and **Lw**) as index terms showed no significant differences in average precision (< 5.0%). That was not expected, since it is well-known that stemming improves performance in retrieval environments. However, this does not seem to be the case in classification environments. Classifiers can been seen as long queries. While retrieval queries (especially in Web environments) contain usually 2-3 keywords, the average length of our classifiers for these experiments were 28.9, 26.1, and 26.1 keywords respectively. An automated method for

[4] Specifically, we called the `morphstr()` function which tries to find the base-form (lemma) of a word or collocation, given its part-of-speech. WordNet is created by Cognitive Science Laboratory, Princeton University, 221 Nassau St., Princeton, NJ 08542. It is available for anonymous ftp from `clarity.Princeton.edu` and `ftp.ims.uni-Stuttgart.de`.

building classifiers like Rocchio's, given sufficient training data, will identify and include all potential morphological variants of significant keywords into a classifier. This makes any form of morphological normalization in such environments redundant. Nevertheless, when no sufficient training data are available (like for the small topics), differences in performance grow larger. In this case, lemmatization is slightly better than stemming which is slightly better than no stemming at all.

The results suggest that for short queries (like in text retrieval), or for insufficient training data (like at the beginning of a text filtering task), morphological normalization will be useful, and lemmatization will be more beneficial for effectiveness than stemming since it is less error-prone. For long and more precise queries (like classification queries derived from sufficient training data), morphological normalization has no significant impact on effectiveness. In any case, morphological normalization reduces the number of terms an information seeking system has to deal with, so it can always be used as a feature reduction mechanism.

7.2.4.2 POS-based Indexing

The experiments based on indexing sets derived from combinations of part-of-speech categories (**Ln, Lnj, Lnv**, and **Lnjv**) presented, for large topics, no significant improvements over the baseline of stemmed words. The same argument as above seems to apply here as well: large training sets make linguistic information redundant. For small topics, however, **Ln** and **Lnj** present almost significant (≈ 5.0%) improvements. All these experiments included, at least, the category of nouns. When we tried to exclude nouns, performance degraded greatly, confirming the importance of nouns for indexing.

A weak conclusion can be drawn. The union of nouns and adjectives (**Lnj**) performs best, while the addition of verbs reduces performance, and adverbs do not make a difference (we should remind the reader that the only difference between the indexing sets **Lnjv** and **Lw** is that the latter includes adverbs). The poor performance of verbs may be related to a limited or poor usage of them in the Reuters data, or to some bad interaction between nouns and verbs. A confusion between nouns and verb arises from the fact that most nouns can be verbed (e.g., *verb* → *verbed*) and verbs can be nominalized (e.g., *to visit* → *a visit*). This issue requires a further investigation.

Despite the non-significant differences in average precision, part-of-speech information may be used to assist term selection mechanisms. Table 7.4 gives a comparison of the number of distinct terms our system had to deal with in different experiments. It can

run	distinct terms	reduction
w	34030	baseline
Sw	27205	−20.0%
Lw	29377	−13.7%
Ln	23039	−32.3%
Lnj	26952	−20.8%
Lnv	24997	−26.5%
Lnjv	28804	−15.3%

Table 7.4: Distinct term occurrences.

be seen that the lemmatized union of nouns and adjectives **Lnj** consists of 20.8% less indexing terms than the indexing set of all keywords **w**, while it preserves the effectiveness (it actually improves it for small topics). Such a POS-based feature reduction mechanism has already been seen in (Rüger, 1998) where nouns and adjectives were assumed to be most vital in representing document contents, but no comparative empirical evaluation was given.

7.2.4.3 Composite Indexing Terms

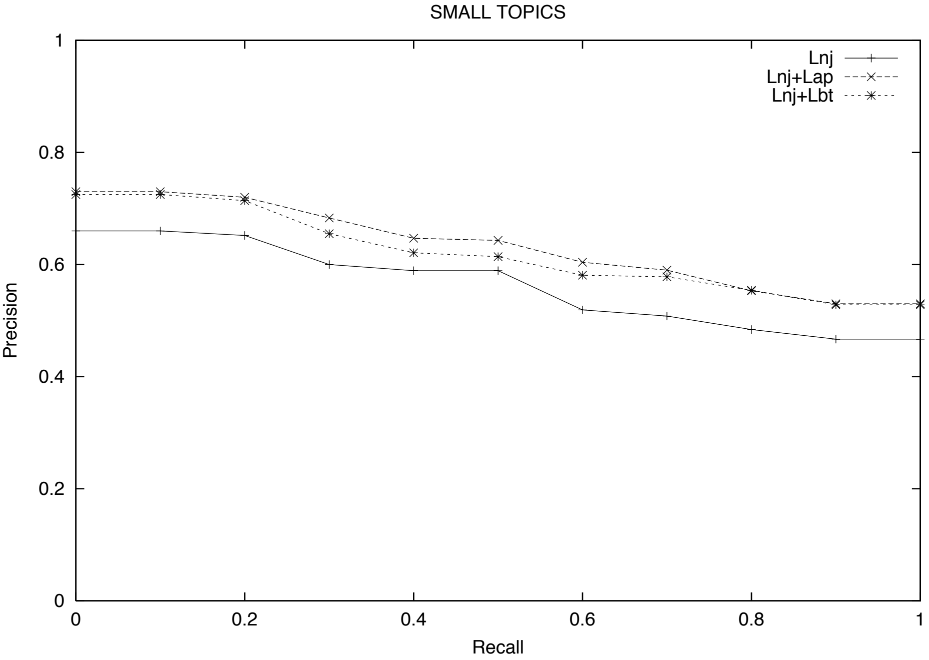
Since the best performance was presented by **Lnj**, we decided to add to this run composite terms in the form of lemmatized adjacent pairs **Lnj+Lap**, or lemmatized binary terms **Lnj+Lbt**.

Both experiments led to significant improvements (> 5.0%) in average precision. Considering **Lnj** as the baseline, the improvement was 12.4% (small topics) and 5.0% (large topics) for adjacent pairs, and 10.1% (small topics) and 5.3% (large topics) for binary terms. Figure 7.1 gives the 11-point interpolated recall-precision curves.

We did not use a special weighting scheme for composite terms. Composite terms were simply mixed up with single terms and weighted using the same *ltc* weighting formula. This clearly violates the *term independence* assumption of the vector space model. In order to compensate for this, when single and composite (phrasal) terms are indexed together, composite terms are traditionally weighted lower (Fuhr et al., 1993), something we did not do. This suggests that there is margin for even better performance assuming a proper weighting scheme.

Unfortunately, binary terms did not prove more effective than adjacent pairs. That was unexpected, since the syntactically canonical nature of binary terms was thought to outperform word adjacency criteria. In a further investigation, we measured how effective the syntactical normalization had been. Figure 7.2 shows the comparative growth of binary terms and adjacent pairs as the dataset grows in documents. In the whole dataset, the total distinct adjacent pairs were 121,185, while the binary terms were 111,631 (7.9% less). Clearly, our syntactical normalization had some effect, but not as extended as we expected.

How limited the syntactic normalization was, is more clear in Figure 7.3. It is well-known that the number of distinct words in a growing document collection grows with the square root of the total number of word occurrences. It is obvious from this figure that this extends also to the subset of **Lnj** for our dataset. One could expect that the same holds for composite terms, but the number of such enriched terms grows even faster. We expected that the syntactically canonical nature of binary terms would have resulted in a less steep curve than that of adjacent pairs, but obviously it did not.



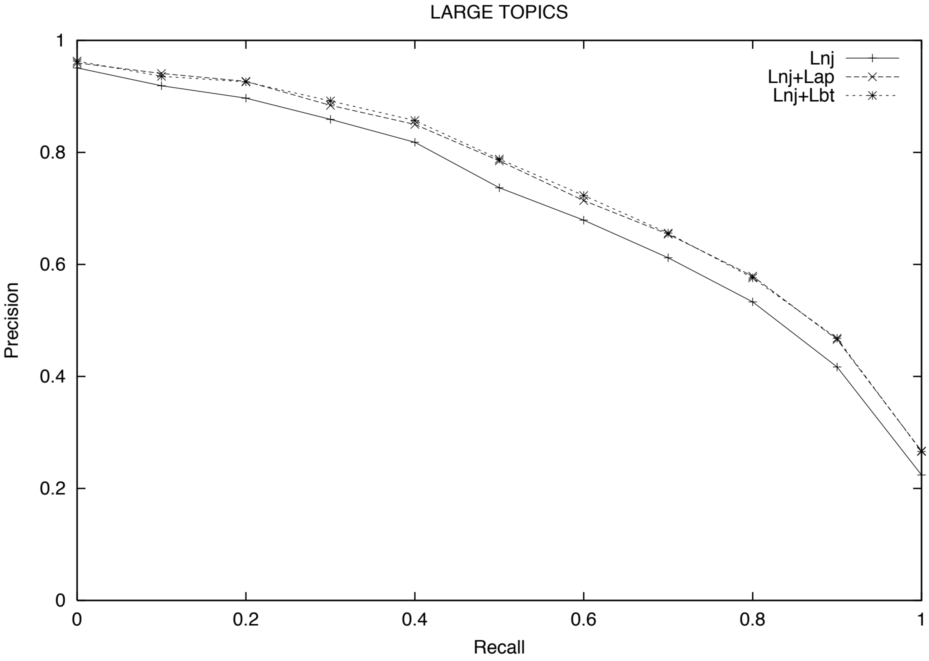


Figure 7.1: The impact of adding composite terms to nouns and adjectives.

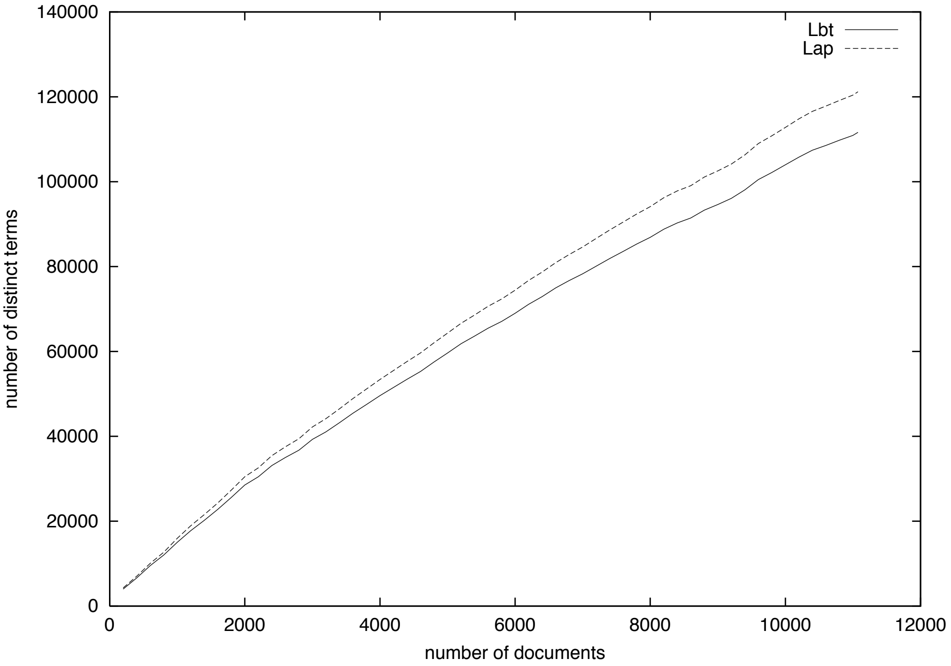


Figure 7.2: Number of distinct terms as a function of the number of documents.

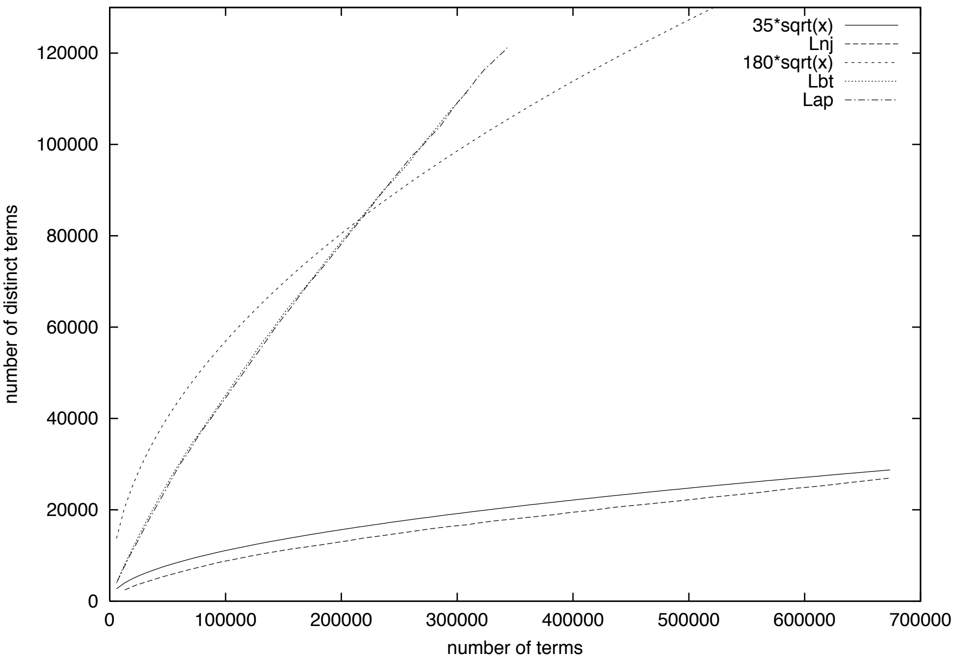


Figure 7.3: Number of distinct terms as a function of the total term occurrences. Two square root curves are shown for comparison purposes. The curves of **Lap** and **Lbt** are overlapping. Obviously, the growth of composite terms cannot be approximated with a square root.

7.3 Filtering and Linguistic Considerations

Our experimental results using linguistically-motivated indexing terms suggest that part-of-speech information is beneficial to indexing. We found that a traditional keyword-based indexing set can be reduced to retain only its nouns and adjectives without hurting effectiveness, even slightly improving it.

Augmenting indexing sets with composite terms resulted in significant improvements in effectiveness for both adjacent pairs and head–modifier pairs. Nevertheless, head–modifier pairs have not proven better than adjacent pairs despite their syntactically canonical nature. The natural language processing techniques used were very limited, but the investigation suggests that using better linguistic tools would improve performance.

A comparison of lemmatization to stemming was not found to produce significant improvements, although lemmatization is considered less error-prone. In fact, both of these forms of morphological normalization were not found to improve significantly the effectiveness of information seeking environments characterized by relatively complete and accurate information needs, such as classification, categorization, or routing given sufficient training data. However, it still seems beneficial for incomplete and imprecise information needs, such as short retrieval queries or near the bootstrapping of filtering tasks. In any case, morphological normalization as much as part-of-speech information may be used to assist feature reduction techniques.

Chapter 8

Conclusions and Further Research

This thesis has mainly dealt with two issues in information seeking:

- document filtering.
- representation of textual information.

In this final chapter, we summarize our approach and results, draw conclusions, and discuss directions for further research.

8.1 Document Filtering

Document filtering is a task which assumes long-term user interests and dynamic information sources. In contrast to traditional information retrieval, documents in filtering are not all available at the time of the initiation of the task, but they arrive one by one or in batches, forming a stream of documents. Therefore, a binary decision should be made for each arriving document whether to retrieve it or not.

Filtering has been traditionally seen as a special case of the information retrieval task. The long term nature of user interests has been explored by *relevance feedback* mechanisms. Binary decisions have been enforced by thresholding the document scores given by the probability of relevance or other similarity measures. However, relevance has usually been assumed to be static over time, the relation between the value of information and its temporal locality has not widely been explored, and not much effort has been put into thresholding which has mostly been realized using *ad hoc* techniques. Our contribution to filtering has been both theoretical and empirical, and concerns these matters.

8.1.1 IF and IR: two sides of the same coin, indeed?

The starting point for our analysis and approach to filtering has been the article of (Belkin and Croft, 1992) on the parallel between filtering and retrieval. Since then, the fact that filtering and retrieval are different has become a well accepted view in the IR community and we believe that our contribution goes definitely beyond that.

In Chapters 1 and 2, we have presented a collection of ideas: a definition of the filtering task, a definition of the filtering topic, two different classifications of topics (one based on relevance and the other on temporal aspects), a classification of adaptivity, and ways of using temporal information for retrieving documents and for feature selection. Moreover, we have discussed potential dangers such as selectivity traps, and paid attention to practical issues such as incrementality.

All these are the result of the bottom-up approach we have followed to deal with filtering during the last few years. Guided by experiments, we have formulated what we believe lies on top and is important for effectiveness as well as for efficiency. As our most recent piece of work, Chapter 2 poses more questions than the answers we provide in the rest of this thesis. Sometimes, however, well posed questions can be as important as answers. Our formalization of the task may prove useful for the design of new approaches.

8.1.2 Time Distributions

In Chapter 3, we have experimented with time distributions. Our hypothesis has been that terms which occur in a temporally-clustered manner correspond to temporal real-world events, thus they are not good predictors for relevance in the future. We have tested the hypothesis in a term selection experiment, eliminating terms whose occurrences are not spread fairly in the time-line or in the sequence of relevant documents. Our results have been inconclusive, although promising. The approach has been a brute-force one, nevertheless, it has resulted in comparable effectiveness to eliminating terms with document frequency thresholding. What has influenced most our approach in proving the hypothesis, has been the dataset used: the Reuters-21578 collection. Its limited time-span gives little scope for temporally local events.

At any rate, the issue of using time distributions in retrieval tasks is not settled. In Chapter 3, we have considered time distributions of terms. Further research should investigate other time distributions as well, such as those of relevant documents introduced in Section 2.3. Certain features of a stream or of training data can be seen as time series. Time series analysis and forecasting techniques may provide additional evidence of relevance to those of traditional time-disregarding retrieval models. The mathematical background is already there; it should only be applied appropriately and evaluated empirically. The most important issue is rather the availability of suitable data for experimentation. In our experience, such approaches require:

1. corpora collected over long periods of time.

2. time-stamped documents.

3. topics with large numbers of relevant documents.

The Reuters-21578 corpus does not satisfy the first requirement, and most of its topics do not conform with the third either. The OHSUMED collection satisfies the first, marginally the second, but not the third one. The lack of datasets suitable for controlled retrieval experiments of this kind is a serious drag.

8.1.3 The S-D Threshold Optimization

In Chapter 4, we have developed a novel method for optimizing thresholds, namely, the *score distributional (S-D) threshold optimization*. The method is capable of optimizing any effectiveness measure defined in terms of the traditional contingency Table 4.1. The analysis we made, we believe, is general enough to apply to a range of retrieval models, from probabilistic to vector space. Moreover, the method can be applied incrementally, a highly desirable feature for adaptive environments.

We have provided a range of choices, from very accurate and computationally expensive to practical and less expensive approximations. Whether the more accurate choices capitalize in improvements in classification effectiveness still remains to be seen. A practical version of the S-D optimization was evaluated in the context of our TREC-9 experimentation and found to be very effective (Chapter 5).

Arguably, the S-D optimization is one of our most important contributions to filtering and to binary classification tasks in general. The optimization is based on the distributions of relevant and non-relevant document scores. Our work in modeling these distributions may prove useful beyond threshold optimization problems. It can be applied to other retrieval environments that may use score distributions, e.g., distributed retrieval (Baumgarten, 1999), or topic detection and tracking (Spitters and Kraaij, 2000).

8.1.4 The Prototype FilterIt System

FilterIt was developed to demonstrate the feasibility of our ideas and evaluate the methods in the TREC-9 Filtering Track (Chapter 5). The system combines all our methods together with other proven techniques in a rather consistent way. Moreover, we have paid special attention to incrementality, minimizing the computational and memory requirements without sacrificing too much accuracy. Let us summarize the techniques incorporated in FilterIt: accurate and incremental adaptivity as soon as a single training document becomes available, local adaptivity, on-the-fly term selection, the S-D threshold optimization, initial query elimination, and query zoning. Moreover, we have empirically determined or at least motivated the effective range of any parameters.

Our first participation to TREC-9, has motivated a great deal of research, experimentation, and triggered new ideas. We did not experiment again with time distributions, however, for the dataset given was not suitable according to the requirements we mentioned above. Instead, we investigated the value of retrieved documents as training examples with respect to their freshness by using local adaptivity. Local adaptivity is necessary when tracking relevance drifts. For this purpose, we introduced the notion of the half life of a training document. The approach has presented promising results, even by using the same half life value for all topics filtered. However, effectiveness seemed to be optimized for considerably different values per topic. Our plans for further research include finding a way of detecting relevance drifts in order to select appropriate half life values.

We have found adaptive filtering an especially sensitive task. What makes it so sensitive is that the system is provided with absolutely no relevance feedback for non-retrieved documents. Any relevance statistics collected in this way are bound to be *partial* in the sense that they do not represent a sample of the whole document space, but a sample of the *retrieved* space, therefore they may be highly misleading. We believe that this problem is important and that further research is required.

Overall, we are very satisfied with our adaptive results in TREC-9; we have clearly achieved the best utility scores in the adaptive and batch-adaptive tasks that we have participated. The approach of combining several techniques worked out well. The FilterIt system is a typical example of: *the whole is more than the sum of its parts*.

8.2 Linguistically Motivated Indexing

In the last decade, the availability of large amounts of digital information, especially textual, has exposed the inadequate effectiveness of keyword-based retrieval models. It has been conjectured many times that a better representation of textual information should go beyond simple keywords, including groups of words (phrases), some form of regularization of words, word order, and meaning. Indeed, many researchers have developed such techniques, but experiments have shown considerable variation in effectiveness making it difficult to establish which techniques actually work and which do not.

In Chapter 6, guided by failures and successes of previous research, we have developed a phrase-based retrieval model which incorporates different kinds of linguistic normalization. Since the goal was to improve retrieval and not to validate some linguistic theory, the suggested scheme does not go into deep linguistic and semantic analysis. What is new in our approach is not the individual techniques, but rather combining them in a coherent and consistent way in a single model, removing details which we believe are unnecessary for retrieval purposes, simplifying the matters greatly.

In Chapter 7, we set out to evaluate several linguistically motivated indexing (LMI) schemes: a part of our suggested model and a few other simple choices. Although, indexing concerns most information seeking tasks, our experiments mainly focused on the performance of LMI in relevance feedback environments. The reason for this was that the general context of the research described in this thesis has been information filtering which is a certain type of relevance feedback environment.

Our empirical evaluation is not complete. We did not manage to evaluate the combined effect of all methods described in in Chapter 6. In retrospect, such a full-scale evaluation had been rather too ambitious. The normalization techniques we have tried do not seem sufficient to improve retrieval effectiveness. More extended normalizations have been developed for the LCS system (Koster et al., 1999), but we are still waiting for their results. Nevertheless, our results so far suggest that part-of-speech information is beneficial to indexing. We found that a traditional keyword-based indexing set can be reduced to retain only its adjectives and nouns without hurting effectiveness, even slightly improving it.

The most important conclusion we can draw is that augmenting indexing sets with composite terms — irrespective of whether these are linguistically canonical or not — improves effectiveness in relevance feedback environments with large training data. Some training/learning methods, given sufficient training data, may be capable of identifying the potential composite terms automatically, making forms of non-extended normalization less important. Nevertheless, when no sufficient training data are available, e.g. for short retrieval queries or near the initiation of a filtering task, some forms of normalization (e.g. morphological) are beneficial.

8.3 Outlook

With this thesis, we have contributed to improving the performance of information seeking, by attacking several small but important problems related to: adaptive document filtering, temporally-dependent data, and linguistically-motivated representations of textual information. We have tackled rather successfully the first two by presenting a useful and unconventional formalization of the problem, practical solutions, and quite promising empirical evaluations. Concerning the representation issue, we have present a model capable of dealing with linguistic variation, however, its validation has not been complete. At any rate, our contribution to information seeking has brought up new interesting questions, and may prove useful for the design of new effective approaches.

In retrospect, assuming that better information seeking means more user satisfaction, perhaps a more radical change in the focus of research is needed. Maybe the future is not to provide better ranking of retrieved documents but to supply the very information a user is seeking. Filtering is a step in this direction, providing the documents that may contain this information, instead of a ranked list. Moreover, a compact summary of retrieval results or a brief answer might be more usable for an average user than sets of documents. Automatic summarization, question answering, and information extraction systems require advanced NLP techniques, however.

Two things seem certain for the future: Information seeking will keep diversifying to serve new information needs of unprecedented nature, and NLP and other linguistic resources will become an indispensable part of effective information seeking systems. With the digital information overload, we are going through a golden age for the related sciences and technologies.

Appendix A

Basic Testbed

A.1 The Vector Space Model

In the *vector space model* (Salton, 1975), documents are represented by weighted vectors, e.g.

$$D = [d_1, \ldots, d_K] \ , \tag{A.1}$$

where d_i is the weight of the ith indexing term for this document, and K indexing terms are used. A user request, called *query* is represented in the same manner, using the same set of indexing terms:

$$Q = [q_1, \ldots, q_K] \ . \tag{A.2}$$

An indexing term may be a word, phrase, or other linguistic entity. We will leave open what a term is, and mention only that indexing terms are assumed to occur in documents in a statistically independent manner; i.e. each indexing term is considered an independent dimension in the K-dimensional indexing space.

A measure of similarity between the document and the query is the cosine of the angle of their vectors, i.e. *cosine similarity*. Assuming that document and query vectors are normalized to unit lengths, the cosine similarity is the dot-product of their vectors:

$$S(D, Q) = \sum_{i=1}^{K} d_i q_i \ . \tag{A.3}$$

Filtering can be done by thresholding the similarity:

$$F(D, Q) = \begin{cases} \text{select } D \ , & \text{if } S(D, Q) > \theta \ . \\ \text{reject } D \ , & \text{otherwise .} \end{cases} \tag{A.4}$$

The original vector space model does not cover several important issues:

- term weighting
- query adaptation
- threshold selection

Next, we will see how term weighting and query adaptation may be performed. We deal with threshold selection in Chapter 4.

A.2 Term Weighting

A.2.1 *ltc*

The *ltc* weighting scheme, commonly used in text retrieval (Buckley et al., 1994), specifies the weight d_i of term i for a document D as

$$d_i = \frac{tf_i \times \log(N/n_i)}{\sqrt{\sum_{j=1}^{K}(tf_j \times \log(N/n_j))^2}} \ , \tag{A.5}$$

where N is the total number of documents, n_i is the number of documents in which term i occurs, and tf_i is

$$tf_i = \begin{cases} 0 \ , & \text{if } f_i = 0 \ . \\ \log(f_i) + 1 \ , & \text{otherwise.} \end{cases} \tag{A.6}$$

f_i is the number of occurrences of the term in D. Query terms are weighted in the same way.

ltc has been found to perform better than other weighting schemes, e.g. *atc*, *lnc*, and *bnn*, in document categorization tasks on the Reuters data (Section A.6.1) and in topic detection and tracking on data from Reuters and CNN (Yang and Pedersen, 1997; Yang et al., 1998). We have moreover tried binary weighting, and tf-thresholding (removing all terms i with $f_i < 2$ per document) before *ltc* weighting, but both have resulted in worse performance than *ltc* in the Reuters-21578 dataset.

A.3 Learning

Filtering involves training data. As documents are filtered, users may explicitly provide relevance judgements for some of the retrieved documents. Moreover, the system may generate judgements inferred from user's behaviour. Non-retrieved documents are not presented to the user, thus no relevance judgments are generated for them. Judged documents may be used as training data to adapt the initial user request in order to achieve better effectiveness in the future.

When training documents are to be filtered for a profile, an ideal query Q_{ideal} should score all relevant above a threshold θ and all non-relevant below. If Q_{ideal} exists, there is an iterative procedure called *fixed-increment error correction* (Nilsson, 1965) which ensures that any initial Q_0 will converge to Q_{ideal} in a finite number of steps:

$$Q_i = \begin{cases} Q_{i-1} + cD \ , & \text{if } S(D, Q_{i-1}) \leq \theta \text{ and } D \in \mathcal{R} \ . \\ Q_{i-1} - cD \ , & \text{if } S(D, Q_{i-1}) > \theta \text{ and } D \in \mathcal{N} \ . \end{cases} \tag{A.7}$$

No changes are made to Q_{i-1} if it classifies correctly. The value of constant c is arbitrary so it is usually set to 1. In practice, it may be necessary to iterate many times over the training set before Q_{ideal} is reached. However, such an ideal query might just not exist; or even if it does exist, it might be overfitting the training documents and not generalize to new documents.

A.4 Rocchio's Relevance Feedback Method

Rocchio's relevance feedback method (Rocchio, 1971) has been developed in the context of the vector space model. Classifiers based on it have proven to be quite effective in filtering and classification tasks (Ittner et al., 1995; Schapire et al., 1998; Ragas and Koster, 1998). It performs well in a situation where only a few training documents are available, see e.g. (Ragas and Koster, 1998), and this is exactly the case in the adaptive filtering task. In such a situation, the initial query becomes important and the method can moreover deal in a suitable way with the topic descriptions.

Rocchio defined the *optimal* query as the one which maximizes the difference between the average score of relevant and the average score of non-relevant documents. Under this definition, Rocchio showed that an optimal query vector is the difference between the average vectors of relevant and non-relevant documents:

$$Q_{\text{Rocchio}} = \frac{1}{|\mathcal{R}|} \sum_{D \in \mathcal{R}} D - \frac{1}{|\mathcal{N}|} \sum_{D \in \mathcal{N}} D \ , \tag{A.8}$$

where $|.|$ denotes the number of documents in a stream. Of course, any other vector cQ_{Rocchio} where c a positive constant is also optimal.

To maintain the focus of a user's initial request Q_0, researchers have found that it is useful to include it in the formula. The version of Rocchio's method traditionally used for relevance feedback is

$$Q_{\text{relfeed}} = \alpha\, Q_0 + \beta \frac{1}{|\mathcal{R}|} \sum_{D \in \mathcal{R}} D - \gamma \frac{1}{|\mathcal{N}|} \sum_{D \in \mathcal{N}} D \ , \tag{A.9}$$

where Q_0 is the initial query, $\mathcal{R}$ and $\mathcal{N}$ are the sets of relevant and non-relevant documents respectively, and $|.|$ denotes the number of elements in a set. The parameters α, β, and γ control the relative contribution of the initial query, and that of the relevant and non-relevant documents to the new query Q. All components which end up with negative weights in Q_{relfeed} are removed.

β and γ control the relative contribution of relevant and non-relevant documents. Usually it is $\beta > \gamma$ because relevant documents are better training examples than non-relevant. Although, relevant documents indicate where the position of a topic in the document space may be, non-relevant documents indicate where this is not without giving a clue of its position. α controls the impact of training documents on the initial query. The larger the α in comparison to β and γ, the more complete and precise the initial query is assumed to be, so only minor modifications due to training are needed. Typical values used for relevance feedback are $\alpha = 2$, $\beta = 4$, and $\gamma = 1$ (Buckley et al., 1994).

Over the years several techniques have been proposed to improve the effectiveness of Rocchio's method. These mainly aim at better term weights in training documents or further optimization of the profile weights proposed by Rocchio's formula, e.g. *dynamic feedback optimization (DFO)* (Buckley and Salton, 1995). Other techniques, e.g. *query zoning* (Singhal et al., 1997), concentrate on selecting a subset of non-relevant documents for training, the ones that have some relationship to user's interest. Sampling the non-relevant document space first to form a query zone and then using only this zone as negative feedback has been found to improve Rocchio's effectiveness.

The query zone is formed as follows. All non-relevant documents are ranked according to their similarity to the initial user request. Then only the top-k most similar are used for training, with typical $k = \max\left(|\mathcal{R}|, \frac{|\mathcal{R} \cup \mathcal{N}|}{c}\right)$, i.e. the query zone size is at least equal to the number of relevant documents and grows with the size of the training stream, where $c = 100$ (Schapire et al., 1998). When using query zones, it has been shown that $\beta = \gamma$ is a reasonable parameter setting (Singhal et al., 1997).

Rocchio's formula does not only modify the weights of the initial query but it also expands it to include new terms. (Buckley et al., 1994) have found that the recall–precision effectiveness increases linearly with the log of the number of terms added, thus massive expansion works out well. Moreover, in the same study it has been found that there is a similar relationship between the log of the number of relevant documents used and the recall–precision effectiveness.

A.5 Effectiveness Measures

In this section, we define the effectiveness measures we use throughout this book. The reader should recall the contingency Table 4.1 and Section 4.2.

A.5.1 Set-based

- **precision:**

$$P = \frac{R_+}{R_+ + N_+} \ . \tag{A.10}$$

- **recall:**

$$R = \frac{R_+}{R_+ + R_-} \ . \tag{A.11}$$

- **F-measure:**

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \ , \quad \beta \in [0, +\infty] \ . \tag{A.12}$$

In Section 5.2.4, we have discussed the evaluation measure trends in the TREC Filtering Track. We have moreover given the definitions of the T9U and T9P measures. T9P is a variant of precision that demands a minimum number of documents to be retrieved, penalizing smaller retrieved sets. T9U is a linear utility function (Sections 4.2.1 and 4.5.3) with a fixed lower bound that ensures that an individual topic which performs really badly will not dominate the average over all topics.

A.5.2 Rank-based

- **average non-interpolated precision:**

  $$\overline{P_{\text{non interpolated}}} = \frac{1}{r}\sum_{i=1}^{r} P_i \ , \tag{A.13}$$

  where P_i is the precision at the position of the ith relevant document in the ranked list.

- **11-point interpolated recall–precision:**

  First, the recall and precision are calculated at every rank of the list. If any relevant documents score zero, they are ranked at the bottom of the list *below* all non-relevant which score zero. Then, the pairs of recall–precision are interpolated at 11 standard recall levels $R_s = s, s = 0, 0.1, \ldots, 1$. We use the interpolation method described in (van Rijsbergen, 1979). According to this method, a set of recall-precision pairs $G = \{(R, P)\}$ is interpolated as:

  $$P_s = \{\max P : R' \geq R_s \ \text{ and } \ (R', P) \in G\} \ , \tag{A.14}$$

  where P_s is the precision at the standard recall level R_s. This interpolation method estimates at R_s the best possible precision achieved by the system.

- **11-point interpolated average precision:**

  $$\overline{P_{\text{11 point interpolated}}} = \frac{1}{11}\sum_{s} P_s \ . \tag{A.15}$$

A.6 Test Collections

In Section 5.2.3, we have described the OHSUMED collection, which was used as test data in the TREC-9 Filtering Track (Chapter 5). We will briefly describe here the Reuters collection, which we have used for our term selection (Chapter 3) and linguistically-motivated indexing (Section 7.2) experiments.

A.6.1 Reuters-21578

The Reuters-21578 (distribution 1.0) text categorization test collection is a resource freely available for research in Information Retrieval, Machine Learning, and other corpus-based research[1].

For our experiments, we produce the Modified Apte (*ModApte*) split (training set: 9,603 documents, test set: 3,299 documents, unused: 8,676 documents). The ModApte split is a subset of the Reuters documents about economic topics such as *income*, *gold*, and *money-supply*. (Hayes and Weinstein, 1990) discuss some examples of the policies

[1] More information, the collection, and its documentation are available from: `http://www.research.att.com/~lewis/`

(not always obvious) used by the human indexers in deciding whether a document belongs to a particular topic category. Documents can be assigned to more than one topic, i.e. they are *multi-classified*. We use only the topics which have at least one relevant training document and at least one relevant test document; these are 90 topics in total.

Appendix B

Threshold Optimization

In this appendix, we investigate whether a Central Limit Theorem (Laha and Rohatgi, 1979) applies to S_m (Equation 4.19) in the limit of a large number of dimensions m, and that the score distribution becomes Gaussian in this limit. If the answer to the question *whether* a Gaussian limit appears is yes, then the next question is *when* it appears, i.e., how soon Gaussian shapes appear as m grows.

First, we[1] prove in Appendix B.1 that a Gaussian limit is not likely for the distribution P_{nr} of non-relevant document scores, and if it appears, then only at a very slow rate with m. In Appendix B.2 we prove that that a Gaussian limit appears for the distribution P_{r} of relevant document scores. Furthermore, we show that the distribution approaches the Gaussian quickly, such that corrections go to zero as $1/m$.

B.1 Non-Gaussian Limit for Non-Relevant

Let C_{nr} be the class of non-relevant documents. In order to investigate the behavior of the score distribution for C_{nr}, we investigate the *cumulants* (Laha and Rohatgi, 1979). These are defined through the moment generating function $\phi(-\imath t)$ of the score distribution, where ϕ is the characteristic function.

$$K_m^{(r)} := \frac{d^r \log \phi(-\imath t)}{dt^r}\bigg|_{t=0} \quad . \tag{B.1}$$

The first cumulant $K_m^{(1)}$ is equal to the mean of the distribution, and the second cumulant $K_m^{(2)}$ is equal to the variance. For a given random variable S_m with given cumulants $K_m^{(r)}$, the cumulants of the variable

$$\hat{S}_m := \frac{S_m - K_m^{(1)}}{\left(K_m^{(2)}\right)^{\frac{1}{2}}} \quad , \tag{B.2}$$

shifted such that it has zero mean and unit variance, are given by

$$\hat{K}_m^{(1)} := 0 \quad , \qquad \hat{K}_m^{(r)} := \frac{K_m^{(r)}}{\left(K_m^{(2)}\right)^{\frac{r}{2}}} \quad r \geq 2 \quad . \tag{B.3}$$

[1] In fact, André van Hameren should be credited for the two proofs in Appendix B.1 and B.2. Without André's knowledge of higher mathematics, these proofs would not have been possible.

For a Gaussian distribution, the logarithm of the moment generating function is given by

$$\log \phi(-\imath t) := tK_m^{(1)} + \frac{t^2}{2} K_m^{(2)} \ . \tag{B.4}$$

The above trivially leads to the conclusion that

Theorem 1 *whenever* $\lim_m \hat{K}_m^{(r)} \to 0$ *for all* $r \geq 3$*, then* $\hat{S} = \lim_m \hat{S}_m$ *is a normal variable, that is, it has a Gaussian probability distribution with zero mean and unit variance.*

This theorem is not the most efficient to prove a Gaussian limit, because it asks for the limiting behavior of *all* cumulants, but it gives a view on how *fast* the limit appears: if the cumulants $\hat{K}_m^{(r)}$ go to zero for large m at a very slow rate, then the probability distribution will start to look Gaussian only for very large m.

Because of the independence assumption, the characteristic function factorizes over the components (Equation 4.21), so that its logarithm becomes a sum over the components of logarithms

$$K_m^{(r)} = \sum_{i=1}^{m} q_i^r \kappa_i^{(r)} \quad , \quad \kappa_i^{(r)} := \frac{d^r \log \phi_i(-\imath t)}{dt^r}\Big|_{t=0} \ . \tag{B.5}$$

The moments of the components depend linearly on the TPs: according to (4.23) we have

$$\mathsf{E}(\omega_i^r) = \int_{-\infty}^{\infty} x^r \, \mathsf{p}_i(dx) = \varepsilon_i \int_{-\infty}^{\infty} x^r \, dF_i(x) \ . \tag{B.6}$$

The cumulants can be written as finite sums of products of the moments, so that in this case $\kappa_i^{(r)}$ is a polynomial in ε_i, i.e.

$$\kappa_i^{(r)} = \varepsilon_i F_i^{(r)} + \mathcal{P}_{2,r}(\varepsilon_i) \quad , \quad F_i^{(r)} := \int_{-\infty}^{\infty} x^r \, dF_i(x) \ , \tag{B.7}$$

where $\mathcal{P}_{2,r}(\varepsilon)$ denotes a polynomial in ε containing orders 2 to r. The interpretation of the cumulants as an expansion in the TPs makes sense, because the TPs are smaller than 1 by definition.

Now, we shall try to derive from the constructed model whether the score distribution converges to a Gaussian for large m, and if it does, what the rate of convergence is. In order to achieve this, we want to replace the sum in Equation B.5 by m times the average query component times the average cumulant. To do this, we need some more assumptions.

The first one is based on the empirical observation that, whereas the TPs ε_i and the moments $\mathsf{E}(\omega_i)$ and $\mathsf{E}(\omega_i^2)$ of the components vary several orders of magnitude, the ratios $\mathsf{E}(\omega_i)/\varepsilon_i$ and $\mathsf{E}(\omega_i^2)/\varepsilon_i$ vary within only one order of magnitude. Together with Equation B.6, the mentioned observation leads to the conclusion that the PDFs F_i do not vary much for the different components, or at least that the variations do not matter much. The important differences between the distributions of the components seems to come from the TPs. We implement this in our model by the assumption that

Assumption 2 *the PDFs F_i are the same for all components, and equal to a single PDF F.*

In order to determine the rate of convergence, we intend to use Theorem 1, so that we need to determine the behavior of the cumulants for large m. According to (B.5) and (B.7), we then need the distributions of the query components (QCs) and the TPs. For both cases, we specify the distribution of the variable by applying a *generalization of Zipf's law.* For QCs let denote

$$\bar{q}_m = \text{the value of the maximal QC.} \tag{B.8}$$

For every m, there is a mapping $\mathcal{Q}_m$ with $\mathcal{Q}_m(1) = 1$, such that the ordered labeling of the variables satisfies

$$q_i = \bar{q}_m \mathcal{Q}_m(i) \quad \text{for every} \quad i = 1, \ldots, m \ . \tag{B.9}$$

Zipf's classical law is obtained with $\mathcal{Q}_m(i) = 1/i$. The distribution of the variable has moments

$$q_m^{(r)} = \frac{\bar{q}_m^r}{m} \mathcal{Q}_m^{(r)} \ , \quad \mathcal{Q}_m^{(r)} := \sum_{i=1}^{m} \mathcal{Q}_m(i)^r \ . \tag{B.10}$$

By definition, the mapping $\mathcal{Q}_m$ is decreasing with $\mathcal{Q}_m(1) = 1$, so that $\mathcal{Q}_m(i)^{r_1} \geq \mathcal{Q}_m(i)^{r_2}$ for all $i = 1, \ldots, m$ if $r_1 < r_2$ and

$$\mathcal{Q}_m^{(r_1)} \geq \mathcal{Q}_m^{(r_2)} \quad \text{for} \quad r_1 < r_2 \ . \tag{B.11}$$

Furthermore, all moments exist, also in the limit of $m \to \infty$, since in the worst case we would have $\mathcal{Q}_m(i) = 1$ for all $i = 1, \ldots, m$, so that $q_m^{(r)} = \bar{q}_m^r$. Therefore, we conclude that

$$\mathcal{Q}_m^{(r)} = \mathcal{O}(m) \quad \text{for all} \quad r > 0 \ , \tag{B.12}$$

where the $\mathcal{O}$-symbol refers to the behavior with m: we say $a_m = \mathcal{O}(b_m)$ if there is a sequence of numbers c_m such that $|a_m/b_m| < c_m$ for all m, and $\lim_{m\to\infty} c_m$ exists. The sums $\mathcal{Q}_m^{(r)}$ do not have to exist in the limit $m \to \infty$: for example in the classical Zipf case, we have $\mathcal{Q}_m^{(1)} = \log m + \mathcal{O}(1)$.

Exactly the same can be done for the TPs, leading to a maximal value $\bar{\varepsilon}_m$, a decreasing mapping $\mathcal{E}_m$ with $\mathcal{E}_m(1) = 1$ and such that $\varepsilon_i = \bar{\varepsilon}_m \mathcal{E}_m(i)$. The moments of the TPs are denoted $\varepsilon_m^{(r)}$.

At this point, we want to notice that the ordering (B.11) of the coefficients $\mathcal{E}_m^{(r)}$ corresponds with the ordering of powers of $\bar{\varepsilon}_m$, which supports the approximation to

Approximation 1 *keep only the lowest order in ε_i for every $i = 1, \ldots, m$ in (B.7),*

since ε_i is smaller than 1.

The following assumption is based on the empirical observation that QCs and TPs seem to take their values independently: if we order the QCs, and make a plot of the values of the corresponding TPs in this ordering, they seem to jump around randomly. This suggests to assume that

Assumption 3 *the TPs and the QCs take their values independently of each other,*

so that the average over the TPs can be taken independently of the average of the QCs. Assumption 3 together with (B.5) and Approximation 1 lead to

$$K_m^{(r)} \overset{m\to\infty}{\longrightarrow} m\, q_m^{(r)} \varepsilon_m^{(1)} F^{(r)} = \frac{1}{m}\, \bar{q}_m^r\, \mathcal{Q}_m^{(r)}\, \bar{\varepsilon}_m \mathcal{E}_m^{(1)}\, F^{(r)} \ . \tag{B.13}$$

The interesting ratio of the cumulants is then given by

$$\frac{K_m^{(r)}}{\left(K_m^{(2)}\right)^{\frac{r}{2}}} = \frac{\mathcal{Q}_m^{(r)}}{\left(\mathcal{Q}_m^{(2)}\right)^{\frac{r}{2}}} \times \left(\frac{m}{\bar{\varepsilon}_m \mathcal{E}_m^{(1)}}\right)^{\frac{r}{2}-1} \times \frac{F^{(r)}}{\left(F^{(2)}\right)^{\frac{r}{2}}} \ . \tag{B.14}$$

According to Theorem 1, the score distribution becomes Gaussian for large m if this final expression vanishes for all $r \geq 3$.

The main use of Assumption 3 is that it enables us to give an estimate of the ratios on the l.h.s. of Equation B.14, in which the contribution of the query completely factorizes from the contribution from the document distribution. A possible difference in the rate of convergence between two document classes only appears in the second and the third factor of the r.h.s. of Equation B.14.

The contribution from the first factor on the r.h.s. of Equation B.14 is determined by the distribution of the QCs. Using (B.11) and the fact that $\mathcal{Q}_m^{(r)} \geq 1$ for every $r \geq 3$, we see that

$$\lim_{m\to\infty} \frac{\mathcal{Q}_m^{(r)}}{\left(\mathcal{Q}_m^{(2)}\right)^{\frac{r}{2}}} = 0 \quad \Longleftrightarrow \quad \lim_{m\to\infty} \frac{1}{\mathcal{Q}_m^{(2)}} = 0 \ . \tag{B.15}$$

So the contribution from the QCs only helps towards a Gaussian limit if $\lim_{m\to\infty} \mathcal{Q}_m^{(2)} = \infty$. We observe a behavior of the distribution of the QCs such that $\mathcal{Q}_m(i) \sim (i)^{-\nu}$ with $0.5 < \nu < 1$. For this behavior, only the case of $\nu = 0.5$ would, strictly speaking, lead to the first factor on the r.h.s. of Equation B.14 to become zero, as $(\log m)^{r/2}$. We conclude that, if the distribution of the QCs helps towards a Gaussian limit, then only very slowly.

The third factor on the r.h.s. of Equation B.14 does not vary with m (by Assumption 2), so that we further only need to look at the second factor, which is determined by the distribution of the TPs. Since $\bar{\varepsilon}_m \leq 1$, only the behavior of the mapping $\mathcal{E}_m$ can help towards a Gaussian limit, and then only if $\lim_{m\to\infty} \mathcal{E}_m^{(1)}$ does not exist. However, we know that $\mathcal{E}_m^{(1)} = \mathcal{O}(m)$, so that the second factor on the r.h.s. of Equation B.14 will never go to zero.

We conclude that it is not likely for C_{nr} to show a Gaussian limit, and if it does, then only at a very slow rate with m.

B.2 Gaussian Limit for Relevant

The analysis for C_{nr} was possible mainly because the TPs were assumed to be small. For the class C_{r} of relevant documents, this does not have to be the case anymore. Actually, the introduction of the TPs does not seem to make sense anymore if they have to be considered close to 1.

For C_{r}, it seems to be more appropriate to adopt the picture of P_m to be centered around a point $\boldsymbol{q}' \in I\!R^m$. It could, for example, have a multi-dimensional Gaussian shape around $\boldsymbol{q}'$, or could be non-zero only inside a hyper-ellipsoid around $\boldsymbol{q}'$ and zero outside. In both examples, the distribution is completely defined by $\boldsymbol{q}'$, and a matrix U_m: for the Gaussian case, it is the variance matrix, and the other case the matrix determines the shape of the ellipsoid. We shall assume that the distribution of C_{r} can be defined by these three elements: the center $\boldsymbol{q}'$, the 'shape'-matrix U_m, and a function that determines the rate of decrease (reasonably fast for a Gaussian, infinitely fast for the ellipsoid, and so on). We summarize the above as

Assumption 4 *in the case of C_{r}, for every m there is an invertible $m \times m$-matrix U_m and a point $\boldsymbol{q}' \in I\!R^m$ such that*

$$\mathsf{P}_m(d\boldsymbol{\omega}) = \frac{|\det U_m|}{\nu_m} f(\,\|U_m(\boldsymbol{\omega} - \boldsymbol{q}')\|^2\,)\, d\boldsymbol{\omega} \quad , \tag{B.16}$$

where $\nu_m := \int_{I\!R^m} f(\,\|\boldsymbol{\omega}\|^2\,) d\boldsymbol{\omega}$ is the volume of the function f in $I\!R^m$, and this function is such that ν_m does not grow faster with m than a power of $m!$.

The factor $|\det U_m|$ is necessary for the correct normalization of the probability distribution in $I\!R^m$. For example with the Gaussian shape, $(U_m^T U_m)^{-1}$ is the variance matrix in this formulation, and $f(x^2) = \exp(-\frac{1}{2}x^2)$. Notice that P_m induces the 'natural' metric $\|\boldsymbol{\omega}\|_{U_m} := \|U_m\boldsymbol{\omega}\|$.

Let us denote

$$V_m := (U_m^{-1})^T \quad \text{and} \quad \alpha_m := \sqrt{2\pi\,\frac{\nu_{m-2}}{\nu_m}} \quad . \tag{B.17}$$

Furthermore, let S_m be the random variable representing the score of documents from C_{r} with probability measure (B.16). We will prove that, under the above assumption,

Theorem 2 *the limiting variable of the sequence*

$$\sigma_m := \alpha_m \, \frac{S_m - \langle \boldsymbol{q}, \boldsymbol{q}' \rangle}{\|V_m \boldsymbol{q}\|}$$

is a normal variable.

Furthermore, we will show that the distribution of σ_m approaches the Gaussian such that corrections go to zero as $1/m$.

We start with expressing $\nu_m := \int_{I\!R^m} f(\,\|\boldsymbol{\omega}\|^2\,) d\boldsymbol{\omega}$ in terms of a one-dimensional integral. This is possible because the integrand only depends on the length of $\boldsymbol{\omega}$, so that we can go over to spherical coordinates and write

$$\nu_m = \gamma_m \int_0^\infty f(x^2) x^{m-1}\, dx \quad , \quad \gamma_m := \frac{2\pi^{\frac{m}{2}}}{\Gamma(\frac{m}{2})} \quad , \tag{B.18}$$

where γ_m is the volume of an m-dimensional sphere with unit radius, and Γ denotes the gamma-function. At this point, we want to note a few facts we shall need later. Firstly, we have

$$\frac{\gamma_{m-1}}{\gamma_{m-3}} = \pi\,\frac{\Gamma(\frac{m-3}{2})}{\Gamma(\frac{m-1}{2})} = \pi\,\frac{\Gamma(\frac{m-3}{2})}{\frac{m-3}{2}\,\Gamma(\frac{m-3}{2})} = \frac{2\pi}{m-3} \quad . \tag{B.19}$$

Secondly, since we demand that ν_m exists for every m, we obviously have

$$\lim_{x\to\infty} f(x^2)x^{m-1} = 0 \quad \text{for every } m \ . \tag{B.20}$$

Thirdly, since we demand that ν_m does not grow faster with m than a power of $m!$, we have

$$\lim_{m\to\infty} \frac{\alpha_{m-2}}{\alpha_m} = \lim_{m\to\infty} \sqrt{\frac{\nu_m}{\nu_{m-2}} \times \frac{\nu_{m-4}}{\nu_{m-2}}} = 1 \ . \tag{B.21}$$

We shall prove the theorem by proving that the moments of the the variables σ_m converge towards the moments of a Gaussian variable. Under the distribution (B.16) the variable σ_m has moments

$$\begin{aligned}\mathsf{E}(\sigma_m^r) &= \int_{\mathbb{R}^m} \left(\alpha_m \frac{\langle \boldsymbol{q},\boldsymbol{\omega}\rangle - \langle \boldsymbol{q},\boldsymbol{q}'\rangle}{\|V_m\boldsymbol{q}\|}\right)^r \mathsf{P}_m(d\boldsymbol{\omega}) \\ &= \frac{\alpha_m^r}{\nu_m} \int_{\mathbb{R}^m} \frac{\langle V_m\boldsymbol{q},\boldsymbol{\omega}\rangle^r}{\|V_m\boldsymbol{q}\|^r} f(\,\|\boldsymbol{\omega}\|^2\,)\,d\boldsymbol{\omega} \ ,\end{aligned} \tag{B.22}$$

where we performed the substitution $\boldsymbol{\omega} \mapsto U_m^{-1}\boldsymbol{\omega} + \boldsymbol{q}'$ on the integration variable. To prove that all moments exist, we can apply the Schwartz inequality, and go over to spherical coordinates to find that

$$\mathsf{E}(|\sigma_m|^r) \le \frac{\alpha_m^r\gamma_m}{\nu_m} \int_0^\infty x^r f(x^2)x^{m-1}\,dx = \alpha_m^r \frac{\nu_{m+r}\gamma_m}{\nu_m\gamma_{m+r}} \ . \tag{B.23}$$

In order to evaluate $(B.22)$ further, we note that every $\boldsymbol{\omega}$ can be written as a linear combination of $\boldsymbol{\omega}_{\mathrm{p}}$ parallel to $V_m\boldsymbol{q}$ and an orthogonal component $\boldsymbol{\omega}_{\mathrm{o}}$, so that $\|\boldsymbol{\omega}\|^2 = \|\boldsymbol{\omega}_{\mathrm{p}}\|^2 + \|\boldsymbol{\omega}_{\mathrm{o}}\|^2$. Furthermore, we can always perform an orthogonal basis transformation such that $\boldsymbol{\omega}_{\mathrm{p}}$ lies along a coordinate axis of $\mathbb{R}^m$, so we can write $\langle V_m\boldsymbol{q},\boldsymbol{\omega}\rangle = \|V_m\boldsymbol{q}\|\omega_{\mathrm{p}}$, and

$$\mathsf{E}(\sigma_m^r) = \frac{\alpha_m^r}{\nu_m} \int_{-\infty}^\infty y^r \int_{\mathbb{R}^{m-1}} f(\,y^2 + \|\boldsymbol{\omega}_{\mathrm{o}}\|^2\,)\,d\boldsymbol{\omega}_{\mathrm{o}}\,dy \ .$$

The integrand is spherical symmetric in $\boldsymbol{\omega}_{\mathrm{o}}$, so that we can go over to spherical coordinates again, and the integral over $\mathbb{R}^{m-1}$ reduces to a one-dimensional integral

$$\mathsf{E}(\sigma_m^r) = \int_{-\infty}^\infty y^r f_m(y)\,dy \ , \tag{B.24}$$

where

$$f_m(y) := \frac{\gamma_{m-1}}{\nu_m\alpha_m} \int_0^\infty f\Big(\frac{y^2}{\alpha_m^2} + x^2\Big)\,x^{m-2}\,dx \ . \tag{B.25}$$

So the moments of the variable σ_m are equal to the moments of a variable with probability density f_m.

We will now prove that the sequence of density functions f_m has a Gaussian limit. Denote the derivative of f by f', then

$$\begin{aligned} \frac{df_m(y)}{dy} &= \frac{\gamma_{m-1}}{\nu_m \alpha_m} \frac{2y}{\alpha_m^2} \int_0^\infty f'\Big(\frac{y^2}{\alpha_m^2} + x^2\Big)\, x^{m-2}\, dx \\ &= -\frac{\gamma_{m-1}}{\nu_m \alpha_m} \frac{y}{\alpha_m^2}(m-3) \int_0^\infty f\Big(\frac{y^2}{\alpha_m} + x^2\Big)\, x^{m-4}\, dx \\ &= -y\, \frac{\alpha_{m-2}}{\alpha_m}\, f_{m-2}\Big(\frac{\alpha_{m-2}}{\alpha_m}\, y\Big) \quad , \end{aligned}$$

where we applied partial integration and used (B.20) in the second step, and used Equation B.19 and the definition of α_m in the last step. Using (B.21), we find that the limiting density f_∞ satisfies the differential equation

$$\frac{df_\infty(y)}{dy} = -y f_\infty(y) \quad ,$$

which has a Gaussian with zero mean and unit variance as solution. Notice that convergence via the differential equation implies *pointwise* convergence, so that we can conclude that the moments $\mathsf{E}(\sigma_m^r)$ become those of a Gaussian distribution with zero mean and unit variance. This then, leads to the conclusion that σ_m becomes a normal variable.

One might argue that f has to be continuous for this proof, for its derivative is used. This derivative, however, only shows up under an integral, so that it is well defined for discontinuous functions with the help of Dirac distributions.

To answer the question how fast the Gaussian limit appears, we just take $\nu_m = a(m!)^k + \mathcal{O}(\,(m!)^k\,)$ with some $a, k > 0$, so that it is easy to see that

$$\frac{\alpha_{m-2}}{\alpha_m} = 1 + \mathcal{O}\Big(\frac{1}{m}\Big) \quad , \tag{B.27}$$

and we can conclude that the distribution converges to the Gaussian as $1/m$.

B.3 Score Distributions and Optimal Threshold

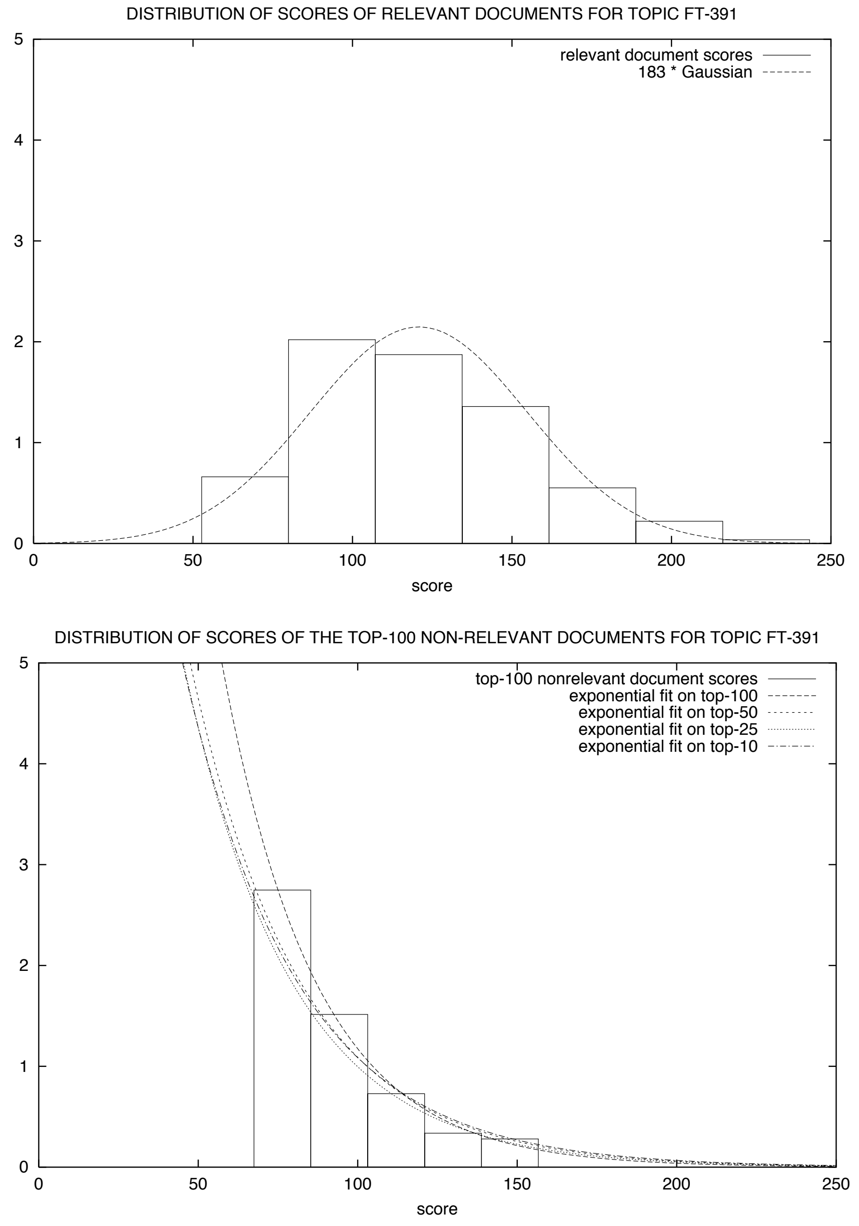


Figure B.1: Empirical and theoretical score distributions.

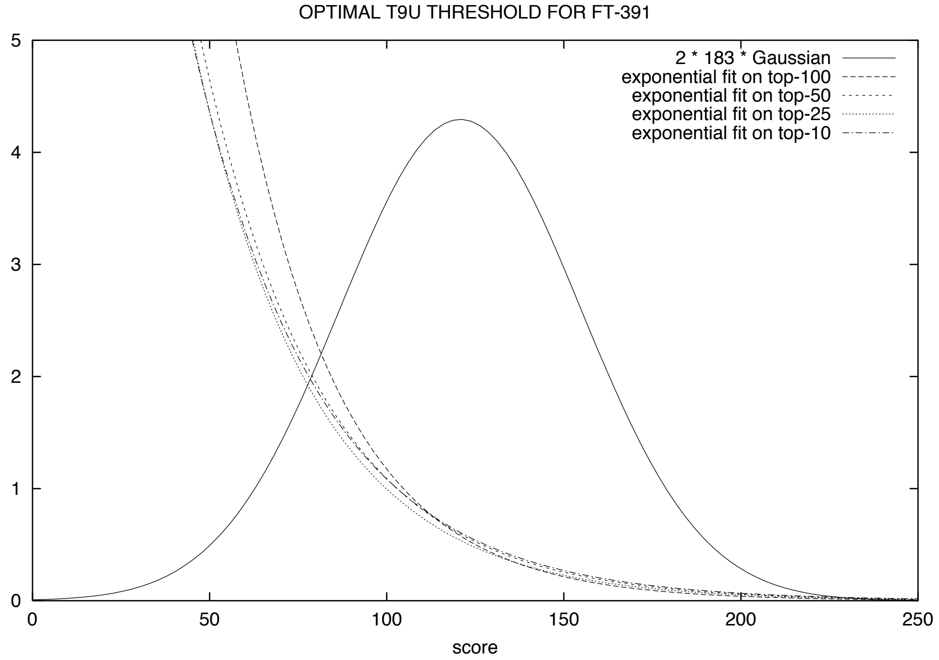


Figure B.2: The optimal T9U threshold.

B.4 Incremental Mean of Scores

Let us assume r documents $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_r$, and a query $\boldsymbol{q}$. The mean score μ_{r} of the documents is:

$$\begin{aligned}
\mu_{\mathrm{r}} &= \frac{1}{r}\sum_{i=1}^{r}\langle \boldsymbol{q}, \boldsymbol{\omega}_i\rangle \\
&= \frac{1}{r}\left(\langle \boldsymbol{q}, \boldsymbol{\omega}_1\rangle + \cdots + \langle \boldsymbol{q}, \boldsymbol{\omega}_r\rangle\right) \\
&= \frac{1}{r}\left(\sum_j q_j \omega_{1j} + \cdots + \sum_j q_j \omega_{rj}\right) \\
&= \frac{1}{r}\sum_j q_j \left(\omega_{1j} + \cdots + \omega_{rj}\right) \\
&= \frac{1}{r}\sum_j q_j \sum_{i=1}^{r} \omega_{ij} \\
&= \frac{1}{r}\langle \boldsymbol{q}, \sum_{i=1}^{r}\boldsymbol{\omega}_i\rangle \; .
\end{aligned}$$

Obviously, the sum of the document tuples is sufficient for calculating the mean score, consequently the individual document tuples can be discarded, i.e. no document buffers are necessary.

B.5 Incremental Mean of Squared Scores

Let us assume r documents $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_r$, and a query $\boldsymbol{q}$. The mean of the squared scores $\mu_{\mathrm{r}}^{(2)}$ of the documents is:

$$\begin{aligned}
\mu_{\mathrm{r}}^{(2)} &= \frac{1}{r}\sum_{i=1}^{r}\langle \boldsymbol{q}, \boldsymbol{\omega}_i\rangle^2 \\
&= \frac{1}{r}\left(\langle \boldsymbol{q}, \boldsymbol{\omega}_1\rangle^2 + \cdots + \langle \boldsymbol{q}, \boldsymbol{\omega}_r\rangle^2\right) \\
&= \frac{1}{r}\left(\sum_j q_j\omega_{1j}\sum_k q_k\omega_{1k} + \cdots + \sum_j q_j\omega_{rj}\sum_k q_k\omega_{rk}\right) \\
&= \frac{1}{r}\left(\sum_{jk} q_j q_k\omega_{1j}\omega_{1k} + \cdots + \sum_{jk} q_j q_k\omega_{rj}\omega_{rk}\right) \\
&= \frac{1}{r}\sum_{jk} q_j q_k\left(\omega_{1j}\omega_{1k} + \cdots + \omega_{rj}\omega_{rk}\right) \\
&= \frac{1}{r}\sum_{jk} q_j q_k\sum_{i=1}^{r}\omega_{ij}\omega_{ik} \\
&= \frac{1}{r}\sum_{jk} q_j\left(\sum_{i=1}^{r}\omega_{ij}\omega_{ik}\right) q_k \,.
\end{aligned}$$

Like the case of the mean score, the individual documents are not necessary for calculating the mean of the squared scores. The sum between the parentheses can be represented by a 2-dimensional matrix which can be updated incrementally when new documents arrive.

Appendix C

TREC-9 Filtering Results

In this appendix, we provide the official TREC-9 evaluation tables of our submitted runs. The results are presented per topic, as well as, averaged over all topics.

The column labeled `#rel` give the total number of relevant documents per topic in the OHSUMED 1988–1991 collection (the test stream). The `min`, `med`, and `max` columns give respectively the minimum, median, and maximum score of all runs submitted by any participant for the corresponding tasks.

In Appendix C.3, the two rightmost columns labeled as `FilterIt-b` and `FilterIt-ba` are not official; they correspond to two *post factum* runs we made, and they are added to the official table for comparison purposes. The same holds for the column `FilterIt-r` in Appendix C.4.

C.1 Adaptive — OHSU topics, Eval: T9U

		KUNa1T9U	KUNa2T9U		All Results	
Topic	#rel	score	score	min	med	max
OHSU1	44	37	25	-100	-15	44
OHSU2	44	5	0	-27	-1	11
OHSU3	165	129	170	-100	31	170
OHSU4	12	5	5	-100	-7	6
OHSU5	44	30	27	-100	-2	40
OHSU6	27	-3	-3	-100	-33	-1
OHSU7	19	7	8	-100	6	13
OHSU8	11	-1	-1	-100	-31	-1
OHSU9	44	-3	-4	-100	-11	5
OHSU10	19	-5	-4	-63	-10	1
OHSU11	84	48	44	-100	5	48
OHSU12	7	-3	-3	-100	-12	-2
OHSU13	77	-1	-1	-100	-4	4
OHSU14	44	-2	-2	-98	-12	8
OHSU15	36	8	3	-100	-12	9
OHSU16	49	10	14	-22	-5	14
OHSU17	27	10	8	-100	-16	10
OHSU18	99	31	22	-100	0	31
OHSU19	95	79	123	-100	20	133
OHSU20	19	13	12	-79	-1	13
OHSU21	112	-1	-1	-100	-7	9
OHSU22	19	-2	-3	-100	-9	1
OHSU23	78	84	84	-45	31	87
OHSU24	74	48	49	-100	7	49
OHSU25	44	-6	-7	-100	-9	-1
OHSU26	43	55	57	-100	6	57
OHSU27	3	-34	-22	-100	-16	0
OHSU28	48	36	39	-36	-3	39
OHSU29	95	56	59	-21	4	59
OHSU30	172	115	80	-84	-4	115
OHSU31	55	-4	1	-71	-3	6
OHSU32	59	-2	-3	-100	-6	0
OHSU33	145	58	24	-100	0	58
OHSU34	12	-2	-2	-100	-9	-1
OHSU35	110	57	88	0	51	103
OHSU36	54	24	11	-100	-8	24
OHSU37	75	33	50	-100	-2	50
OHSU38	62	51	44	-100	28	58
OHSU39	127	-3	2	-100	-6	2
OHSU40	46	17	26	-100	1	29
OHSU41	19	-3	-3	-100	-5	2
OHSU42	53	4	-1	-100	-5	4
OHSU43	60	51	52	-100	18	95
OHSU44	42	-1	-2	-100	-4	19
OHSU45	15	-5	-5	-100	-10	0
OHSU46	25	6	7	-28	-4	7
OHSU47	45	0	0	-23	-1	11
OHSU48	69	-2	-2	-100	-7	12
OHSU49	75	-2	-2	-100	-31	15
OHSU50	27	-3	-3	-100	-11	0
OHSU51	57	-2	-2	-100	-9	13
OHSU52	13	6	2	-100	-32	6
OHSU53	3	-1	-1	-100	-8	0
OHSU54	12	-2	-2	-100	-8	4
OHSU55	30	9	3	-100	-5	9
OHSU56	24	13	11	-100	-6	20
OHSU57	26	13	16	-100	-9	16
OHSU58	10	-2	0	-100	-4	3
OHSU59	45	7	11	-77	-2	19
OHSU60	44	-4	-4	-100	-9	2
OHSU61	26	6	2	-29	-3	6
OHSU62	23	-3	-3	-100	-10	-3
OHSU63	63	-3	0	-100	-7	5
Average		16.8	17.3			

C.2 Adaptive — OHSU topics, Eval: T9P

		KUNa2T9P	KUNa1T9P	All Results		
Topic	#rel	score	score	min	med	max
OHSU1	44	0.360	0.327	0.032	0.380	0.560
OHSU2	44	0.180	0.314	0.120	0.351	0.442
OHSU3	165	0.670	0.714	0.100	0.655	0.779
OHSU4	12	0.040	0.135	0.000	0.103	0.182
OHSU5	44	0.160	0.333	0.081	0.340	0.549
OHSU6	27	0.040	0.040	0.033	0.075	0.119
OHSU7	19	0.180	0.157	0.000	0.232	0.353
OHSU8	11	0.020	0.020	0.000	0.002	0.111
OHSU9	44	0.100	0.173	0.000	0.175	0.351
OHSU10	19	0.080	0.080	0.000	0.060	0.151
OHSU11	84	0.551	0.481	0.074	0.440	0.551
OHSU12	7	0.020	0.000	0.000	0.016	0.096
OHSU13	77	0.140	0.260	0.020	0.236	0.340
OHSU14	44	0.240	0.200	0.000	0.188	0.333
OHSU15	36	0.160	0.160	0.079	0.200	0.273
OHSU16	49	0.120	0.196	0.000	0.246	0.415
OHSU17	27	0.120	0.100	0.033	0.154	0.240
OHSU18	99	0.420	0.468	0.000	0.268	0.468
OHSU19	95	0.691	0.640	0.000	0.640	0.791
OHSU20	19	0.180	0.180	0.000	0.180	0.315
OHSU21	112	0.320	0.526	0.046	0.220	0.526
OHSU22	19	0.200	0.212	0.000	0.138	0.212
OHSU23	78	0.520	0.653	0.363	0.660	0.836
OHSU24	74	0.426	0.533	0.140	0.463	0.643
OHSU25	44	0.000	0.039	0.000	0.001	0.259
OHSU26	43	0.420	0.439	0.123	0.411	0.580
OHSU27	3	0.000	0.000	0.000	0.000	0.039
OHSU28	48	0.260	0.400	0.000	0.300	0.500
OHSU29	95	0.580	0.481	0.217	0.453	0.580
OHSU30	172	0.580	0.593	0.079	0.500	0.593
OHSU31	55	0.140	0.250	0.000	0.275	0.365
OHSU32	59	0.080	0.160	0.000	0.060	0.160
OHSU33	145	0.507	0.439	0.077	0.403	0.580
OHSU34	12	0.000	0.000	0.000	0.060	0.111
OHSU35	110	0.727	0.732	0.000	0.727	0.780
OHSU36	54	0.360	0.356	0.040	0.220	0.360
OHSU37	75	0.520	0.411	0.000	0.370	0.520
OHSU38	62	0.540	0.526	0.180	0.540	0.740
OHSU39	127	0.240	0.220	0.045	0.200	0.340
OHSU40	46	0.320	0.396	0.020	0.377	0.509
OHSU41	19	0.040	0.060	0.000	0.074	0.216
OHSU42	53	0.080	0.080	0.000	0.140	0.258
OHSU43	60	0.680	0.618	0.168	0.618	0.704
OHSU44	42	0.100	0.080	0.080	0.135	0.420
OHSU45	15	0.000	0.040	0.000	0.020	0.060
OHSU46	25	0.080	0.196	0.000	0.080	0.211
OHSU47	45	0.200	0.216	0.000	0.240	0.306
OHSU48	69	0.180	0.431	0.000	0.200	0.462
OHSU49	75	0.160	0.203	0.056	0.160	0.375
OHSU50	27	0.140	0.180	0.101	0.155	0.283
OHSU51	57	0.098	0.140	0.002	0.180	0.386
OHSU52	13	0.060	0.100	0.022	0.080	0.100
OHSU53	3	0.000	0.000	0.000	0.000	0.027
OHSU54	12	0.080	0.137	0.000	0.109	0.180
OHSU55	30	0.040	0.100	0.000	0.180	0.368
OHSU56	24	0.240	0.176	0.045	0.138	0.333
OHSU57	26	0.260	0.216	0.036	0.236	0.360
OHSU58	10	0.040	0.020	0.000	0.060	0.143
OHSU59	45	0.280	0.302	0.000	0.302	0.400
OHSU60	44	0.212	0.258	0.000	0.180	0.340
OHSU61	26	0.160	0.160	0.040	0.160	0.193
OHSU62	23	0.080	0.059	0.000	0.073	0.160
OHSU63	63	0.098	0.140	0.000	0.098	0.200
Average		0.231	0.258			

C.3 Batch — OHSU topics, Eval: T9U

		KUNbaT9U	KUNb	All Results			FilterIt-b	FilterIt-ba
Topic	#rel	score	score	min	med	max	score	score
OHSU1	44	25	17	0	17	27	11	28
OHSU2	44	16	11	0	11	16	8	26
OHSU3	165	174	120	-3	120	174	139	177
OHSU4	12	3	-2	-2	2	4	3	2
OHSU5	44	22	23	0	22	28	10	28
OHSU6	27	-21	-10	-21	0	0	-11	-6
OHSU7	19	12	6	-3	6	16	13	13
OHSU8	11	-29	-7	-29	-1	0	-10	-8
OHSU9	44	-10	-55	-55	-3	0	-20	-4
OHSU10	19	-16	-5	-16	0	0	1	-5
OHSU11	84	44	17	0	28	46	68	45
OHSU12	7	-13	1	-13	0	1	-1	0
OHSU13	77	16	-17	-17	4	16	1	-6
OHSU14	44	25	-1	-1	1	25	3	12
OHSU15	36	1	-5	-5	-2	1	4	4
OHSU16	49	1	-30	-30	1	17	0	-7
OHSU17	27	8	-18	-18	0	8	0	5
OHSU18	99	16	-31	-31	6	19	-42	9
OHSU19	95	125	100	-4	65	125	61	131
OHSU20	19	5	6	0	5	7	8	7
OHSU21	112	66	20	-1	2	66	62	74
OHSU22	19	-2	0	-2	0	1	0	-4
OHSU23	78	91	16	0	16	91	72	89
OHSU24	74	51	20	-3	20	51	40	57
OHSU25	44	-12	-19	-19	0	0	-5	-3
OHSU26	43	51	29	0	20	51	6	51
OHSU27	3	-12	-15	-56	-12	0	-1	0
OHSU28	48	49	12	-1	1	49	1	26
OHSU29	95	44	25	0	4	44	20	45
OHSU30	172	100	23	0	24	100	60	92
OHSU31	55	-2	2	-3	2	11	-8	-8
OHSU32	59	-35	-34	-35	-20	-6	-12	-16
OHSU33	145	62	53	-63	42	62	74	61
OHSU34	12	-3	-74	-74	-1	0	0	-4
OHSU35	110	122	95	-100	95	122	124	143
OHSU36	54	6	12	-3	0	12	3	-10
OHSU37	75	42	13	-1	13	42	19	28
OHSU38	62	62	44	0	44	62	51	59
OHSU39	127	2	-6	-6	-1	9	4	22
OHSU40	46	22	23	-1	22	26	35	32
OHSU41	19	1	-1	-1	0	1	0	0
OHSU42	53	-5	0	-5	0	7	0	0
OHSU43	60	62	44	-3	44	62	64	72
OHSU44	42	12	6	-3	1	12	5	7
OHSU45	15	-9	0	-10	0	0	0	-1
OHSU46	25	3	5	-2	2	5	10	6
OHSU47	45	5	12	-7	-2	12	-9	-7
OHSU48	69	35	2	-1	5	35	22	47
OHSU49	75	14	-4	-8	-1	14	23	-3
OHSU50	27	-24	-5	-24	0	5	-16	-24
OHSU51	57	3	1	-7	1	8	6	9
OHSU52	13	-12	-18	-18	-1	0	6	1
OHSU53	3	-6	-18	-18	-3	0	-8	-3
OHSU54	12	1	-5	-5	0	7	0	-7
OHSU55	30	13	-11	-11	0	13	23	18
OHSU56	24	0	8	-2	0	8	0	8
OHSU57	26	16	15	0	11	16	0	0
OHSU58	10	-2	-31	-31	0	0	0	2
OHSU59	45	8	8	0	7	8	12	10
OHSU60	44	-3	-16	-16	-2	0	4	13
OHSU61	26	5	1	0	0	5	-2	6
OHSU62	23	-4	-13	-13	0	1	0	3
OHSU63	63	-1	-21	-21	0	0	0	0
Average		19.4	5.0				14.8	21.3

C.4 Routing — OHSU topics, Eval: Av. Prec.

		KUNr1	KUNr2		All Results		FilterIt-r
Topic	#rel	score	score	min	med	max	score
OHSU1	44	0.386	0.431	0.000	0.386	0.716	0.4660
OHSU2	44	0.302	0.299	0.000	0.302	0.547	0.5327
OHSU3	165	0.593	0.603	0.000	0.556	0.682	0.7074
OHSU4	12	0.303	0.296	0.000	0.286	0.684	0.4024
OHSU5	44	0.391	0.386	0.000	0.416	0.654	0.5681
OHSU6	27	0.048	0.057	0.000	0.048	0.118	0.1118
OHSU7	19	0.322	0.338	0.000	0.322	0.681	0.6017
OHSU8	11	0.013	0.015	0.000	0.015	0.124	0.0161
OHSU9	44	0.095	0.081	0.000	0.095	0.495	0.1851
OHSU10	19	0.142	0.146	0.000	0.084	0.154	0.1323
OHSU11	84	0.243	0.249	0.000	0.243	0.524	0.6225
OHSU12	7	0.379	0.309	0.000	0.124	0.379	0.1533
OHSU13	77	0.188	0.203	0.000	0.192	0.362	0.3320
OHSU14	44	0.070	0.058	0.000	0.207	0.576	0.2758
OHSU15	36	0.013	0.010	0.000	0.171	0.282	0.3583
OHSU16	49	0.181	0.209	0.000	0.181	0.383	0.2974
OHSU17	27	0.013	0.012	0.000	0.116	0.317	0.2979
OHSU18	99	0.237	0.185	0.000	0.188	0.344	0.3375
OHSU19	95	0.676	0.676	0.000	0.676	0.791	0.7738
OHSU20	19	0.364	0.348	0.000	0.364	0.615	0.4597
OHSU21	112	0.359	0.310	0.000	0.310	0.380	0.5028
OHSU22	19	0.022	0.017	0.000	0.059	0.439	0.1860
OHSU23	78	0.371	0.392	0.000	0.371	0.813	0.6842
OHSU24	74	0.308	0.344	0.000	0.344	0.600	0.6381
OHSU25	44	0.027	0.031	0.000	0.114	0.176	0.1299
OHSU26	43	0.514	0.496	0.000	0.496	0.776	0.6888
OHSU27	3	0.010	0.010	0.000	0.009	0.185	0.0142
OHSU28	48	0.329	0.361	0.000	0.256	0.532	0.3424
OHSU29	95	0.314	0.303	0.000	0.303	0.534	0.3408
OHSU30	172	0.204	0.230	0.000	0.230	0.441	0.4737
OHSU31	55	0.162	0.131	0.000	0.131	0.352	0.3552
OHSU32	59	0.037	0.026	0.000	0.037	0.234	0.1223
OHSU33	145	0.320	0.312	0.000	0.312	0.510	0.4678
OHSU34	12	0.024	0.011	0.000	0.036	0.099	0.0567
OHSU35	110	0.692	0.598	0.000	0.648	0.748	0.7831
OHSU36	54	0.338	0.333	0.000	0.247	0.431	0.1939
OHSU37	75	0.278	0.280	0.000	0.278	0.533	0.5275
OHSU38	62	0.628	0.624	0.000	0.576	0.726	0.6912
OHSU39	127	0.080	0.078	0.000	0.080	0.234	0.2539
OHSU40	46	0.478	0.459	0.000	0.453	0.561	0.6016
OHSU41	19	0.241	0.269	0.000	0.076	0.269	0.1935
OHSU42	53	0.006	0.006	0.000	0.048	0.243	0.2115
OHSU43	60	0.676	0.679	0.000	0.679	0.833	0.7729
OHSU44	42	0.267	0.207	0.000	0.263	0.364	0.4048
OHSU45	15	0.042	0.039	0.000	0.042	0.133	0.0576
OHSU46	25	0.128	0.129	0.000	0.129	0.478	0.4096
OHSU47	45	0.399	0.392	0.000	0.344	0.399	0.3682
OHSU48	69	0.226	0.260	0.000	0.154	0.574	0.4280
OHSU49	75	0.120	0.111	0.000	0.111	0.381	0.3198
OHSU50	27	0.112	0.118	0.000	0.118	0.301	0.2175
OHSU51	57	0.082	0.070	0.000	0.114	0.541	0.4214
OHSU52	13	0.041	0.028	0.000	0.179	0.309	0.3258
OHSU53	3	0.003	0.002	0.000	0.006	0.170	0.1085
OHSU54	12	0.542	0.556	0.000	0.347	0.556	0.4084
OHSU55	30	0.304	0.348	0.000	0.136	0.458	0.5460
OHSU56	24	0.315	0.255	0.000	0.153	0.322	0.4457
OHSU57	26	0.414	0.415	0.000	0.376	0.510	0.3684
OHSU58	10	0.008	0.008	0.000	0.008	0.517	0.4874
OHSU59	45	0.254	0.232	0.000	0.232	0.542	0.3507
OHSU60	44	0.101	0.117	0.000	0.101	0.266	0.3780
OHSU61	26	0.177	0.190	0.000	0.177	0.382	0.3052
OHSU62	23	0.024	0.039	0.000	0.039	0.306	0.1939
OHSU63	63	0.025	0.029	0.000	0.016	0.112	0.0981
Average		0.237	0.234				0.3731

Bibliography

Alexander, L. G. (1988). *Longman English Grammar.* Longman Group UK Limited.

Allan, J. (1996). Incremental Relevance Feedback for Information Filtering. In Frei, H.-P., Harman, D., Schauble, P., and Wilkinson, R., editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 270–278, Zurich, Switzerland. ACM Press.

Allan, J., Carbonell, J., Doddington, G., Yamron, J., and Yang, Y. (1998). Topic Detection and Tracking Pilot Study — Final Report. In *Proceedings of the Broadcast News Transcription and Understranding Workshop.*

Arampatzis, A., Beney, J., Koster, C. H. A., and van der Weide, T. P. (2000a). Incrementality, Half-Life, and Threshold Optimization, for Adaptive Document Filtering. In Voorhees, E. M. and Harman, D. K., editors, *The Nineth Text REtrieval Conference (TREC-9)*, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST).

Arampatzis, A. and Tsoris, T. (1996). A Linguistic Approach to Information Retrieval. Master's thesis, Department of Computer Engineering and Informatics, University of Patras, Patras, Greece.

Arampatzis, A., Tsoris, T., and Koster, C. (1997a). Irena: Information Retrieval Engine based on Natural language Analysis. In *Proceedings of RIAO'97 Computer-Assisted Information Searching on Internet*, pages 159–175, McGill University, Montreal, Canada.

Arampatzis, A., Tsoris, T., Koster, C., and van der Weide, T. (1998). Phrase Based Information Retrieval. *Information Processing & Management*, 34(6):693–707.

Arampatzis, A., van der Weide, T., Koster, C., and van Bommel, P. (2000b). Linguistically Motivated Information Retrieval. In Kent, A., editor, *Encyclopedia of Library and Information Science*, volume 69. Marcel Dekker, Inc., New York, Basel.
Also available from `http://www.cs.kun.nl/∼avgerino`.

Arampatzis, A., van der Weide, T., Koster, C., and van Bommel, P. (2000c). Term Selection for Filtering based on Distribution of Terms over Time. In *Proceedings of RIAO'2000 Content-Based Multimedia Information Access*, pages 1221–1237, Collège de France, Paris, France.

Arampatzis, A. and van der Weide, T. P. (2001). Document Filtering as an Adaptive and Temporally-dependent Process. In *Proceedings of the BCS-IRSG European Colloquium on IR Research*, Darmstadt, Germany.
Also available from `http://www.cs.kun.nl/∼avgerino`.

Arampatzis, A., van der Weide, T. P., van Bommel, P., and Koster, C. H. A. (1997b). Linguistic Variation in Information Retrieval and Filtering. In de Bra, P., editor, *Informatiewetenschap 1997*, pages 7–10, Eindhoven University of Technology, Eindhoven, The Netherlands.

Arampatzis, A., van der Weide C.H.A. Koster, T., and van Bommel, P. (2000d). An Evaluation of Linguistically-motivated Indexing Schemes. In *Proceedings of the BCS-IRSG, 22nd Annual Colloquium on Information Retrieval Research*, pages 34–45, Sindney Sussex College, Cambridge, England.
Also available from `http://www.cs.kun.nl/∼avgerino`.

Arampatzis, A. and van Hameren, A. (2001). The Score-Distributional Threshold Optimization for Adaptive Binary Classification Tasks. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans. ACM Press. To appear.

Baumgarten, C. (1999). A Probabilistic Solution to the Fusion Problem in Distributed Information Retrieval. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press.

Belkin, N. J. and Croft, W. B. (1992). Information Filtering and Information Retrieval: Two Sides of the Same Coin? *Communications of the ACM*, 35(12):29–38.

Brill, E. (1994). Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, Wa.

Bruza, P. and Weide, T. v. d. (1992). Stratified Hypermedia Structures for Information Disclosure. *The Computer Journal*, 35(3):208–220.

Bruza, P. D. (1993). *Stratified Information Disclosure: A Synthesis between Information Retrieval and Hypermedia*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands.

Bruza, P. D. and IJdens, J. J. (1994). Efficient Probabilistic Inference through Index Expression Belief Networks. In *Proceedings of the Seventh Australian Joint Conference on Artificial Intelligence (AI94)*, pages 592–599. World Scientific.

Buckley, C. and Salton, G. (1995). Optimizing of Relevance Feedback Weights. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Buckley, C., Salton, G., and Allan, J. (1994). The Effect of Adding Relevance Information in a Relevance Feedback Environment. In Croft, W. B. and van Rijsbergen, C. J., editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 292–300, Dublin, Ireland. ACM Press.

Callan, J. (1998). Learning While Filtering Documents. In Croft, W. B., Moffat, A., van Rijsbergen, C. J., Wilkinson, R., and Zobel, J., editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 224–231, Melbourne, Australia. ACM Press, New York.

Croft, W. B. (1993). Knowledge-based and Statistical Approaches to Text Retrieval. *IEEE Expert*, 8(2):8–12.

Dagan, I., Karov, Y., and Roth, D. (1997). Mistake-driven Learning in Text Categorization. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *American Society of Information Science*, 1(6):391–407.

Denning, P. J. (1982). Electronic Junk. *Communications of the ACM*, 25(3):163–165.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley-Interscience, New York.

Dumais, S. (1998). Using support vector machines for text categorization. *IEEE Intelligent Systems*, 13(4):21–23.

Fagan, J. L. (1987). *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods*. PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York 14853-7501.

Fuhr, N., Pfeifer, U., Bremkamp, C., and Pollmann, M. (1993). Probalistic Learning Approaches for Indexing and Retrieval with the TREC-2 Collection. In Harman, D. K., editor, *The Second Text REtrieval Conference (TREC-2)*, pages 519–526, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-215.

Gay, L. S. and Croft, W. B. (1990). Interpreting Nominal Compounds for Information Retrieval. *Information Processing & Management*, 26(1):21–38.

Gonzalo, J., Verdejo, F., Chugur, I., and Cigarrán, J. (1998). Indexing with wordnet synsets can improve text retrieval. In *Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, Montreal.

Harman, D. (1991). How Effective is Suffixing? *Journal of the American Society for Information Science*, 42(1):7–15.

Hayes, P. J. and Weinstein, S. P. (1990). CONSTRUE/TIS: a system for content-based indexing of a database of news stories. In *Second Annual Conference on Innovative Applications of Artificial Intelligence*.

Hersh, W. R., Buckley, C., Leone, T. J., and Hickam, D. H. (1994). OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual ACM SIGIR Conference*, pages 192–201.

Hersh, W. R. and Hickam, D. H. (1994). Use of a multi-application computer workstation in a clinical setting. *Bulletin of the Medical Library Association*, 82:382–389.

Hull, D. (1996). Stemming Algorithms — A Case Study for Detailed Evaluation. *Journal of the American Society for Information Science*, 47(1).

Hull, D. A. (1997). The TREC-6 Filtering Track: Description and Analysis. In Voorhees, E. M. and Harman, D. K., editors, *The Sixth Text REtrieval Conference (TREC-6)*, pages 45–68, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-240.

Hull, D. A. (1998). The TREC-7 Filtering Track: Description and Analysis. In Voorhees, E. M. and Harman, D. K., editors, *The Seventh Text REtrieval Conference (TREC-7)*, pages 33–46, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-242.

Hull, D. A., Grefenstette, G., Schulze, B. M., Gaussier, E., Schutze, H., and Pedersen, J. O. (1996). Xerox TREC-5 Site Report: Routing, Filtering, NLP, and Spanish Tracks. In Harman, D. K. and Voorhees, E. M., editors, *The Fifth Text REtrieval Conference (TREC-5)*, pages 167–180, Gaithersburg, Md. 20899. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-238.

Hull, D. A. and Robertson, S. (1999). The TREC-8 Filtering Track Final Report. In Voorhees, E. M. and Harman, D. K., editors, *The Eighth Text REtrieval Conference (TREC-8)*, pages 35–56, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-246.

Ittner, D. J., Lewis, D. D., and Ahn, D. D. (1995). Text Categorization of Low Quality Images. In *Symposium on Document Analysis and Information Retrieval*, pages 301–315, Las Vegas, NV. ISRI; University of Nevada.

Jones, K. S. and Willett, P., editors (1997). *Readings in Information Retrieval.* Academic Press/Morgan Kaufmann.

Knuth, D. E. (1981). *The Art of Computer Programming — Volume 2: Seminumerical Algorithms.* Addison Wesley.

Koster, C., Derksen, C., van de Ende, D., and Potjer, J. (1999). Normalization and Matching in the DORO System. In *Proceedings of the 21st BCS-IRSG Colloquium on Information Retrieval.*

Kraaij, W. and Pohlmann, R. (1996a). Using Linguistic Knowledge in Information Retrieval. Technical Report OTS Working Paper OTS-WP-CL-96-001, OTS, Utrecht, The Netherlands.

Kraaij, W. and Pohlmann, R. (1996b). Viewing Stemming as Recall Enhancement. In Frei, H.-P., Harman, D., Schauble, P., and Wilkinson, R., editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 40–48, Zurich, Switzerland. ACM Press.

Kraaij, W. and Pohlmann, R. (1998). Comparing the Effect of Syntactic vs. Statistical Phrase Indexing Strategies for Dutch. In Nicolaou, C. and Stephanidis, C., editors, *Proceedings of Second European Conference on Research and Advanced Technology for Digital Libraries ECDL'98*, pages 605–614, Heraklion, Crete. Springer-Verlag.

Krovetz, R. (1993). Viewing Morphology as an Inference Process. In Korfhage, R., Rasmussen, E., and Willett, P., editors, *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 191–202, Pittsburgh, PA, USA. ACM Press.

Kuipers, L. and Niederreiter, H. (1974). *Uniform Distribution of Sequences.* Wiley, New York.

Laha, R. G. and Rohatgi, V. K. (1979). *Probability Theory.* John Wiley & Sons, New York.

Lewis, D. D. (1992). Feature Selection and Feature Extraction for Text Categorization. In *Proceedings of Speech and Natural Language workshop, Harriman, New York, February 23–26*, pages 212–217, San Mateo, CA. Morgan Kaufmann.

Lewis, D. D. (1995a). Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 246–255.

Lewis, D. D. (1995b). The TREC-4 Filtering Track. In Harman, D. K., editor, *The Fourth Text REtrieval Conference (TREC-4)*, pages 165–180, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-236.

Lewis, D. D. (1996). The TREC-5 Filtering Track. In Voorhees, E. M. and Harman, D. K., editors, *The Fifth Text REtrieval Conference (TREC-5)*, pages 75–96, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-238.

Lewis, D. D. and Croft, W. B. (1990). Term clustering of syntactic phrases. In *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 385–405. ACM Press.

Lewis, D. D., Schapire, R. E., Callan, J. P., and Papka, R. (1996). Training algorithms for linear text classifiers. In Frei, H.-P., Harman, D., Schauble, P., and Wilkinson, R., editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298–306, Zurich, Switzerland. ACM Press.

Littlestone, N. (1988). Learning Quickly when Irrelevant Attributes Abound: a NewLinear-threshold Algorithm. *Machine Learning*, 2:285–318.

Lovins, J. B. (1968). Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, 11(1):22–31.

Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A., and Cohen, M. D. (1987). Intelligent Information Sharing Systems. *Communications of the ACM*, 30(5):390–402.

Miller, G. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Mitra, M., Buckley, C., Singhal, A., and Cardie, C. (1997). An Analysis of Statistical and Syntactic Phrases. In *Proceedings of RIAO'97 Computer-Assisted Information Searching on Internet*, pages 200–214, McGill University, Montreal, Canada.

Nilsson, N. J. (1965). *Learning Machines — Foundations of Trainable Pattern Classifying Systems*. McGraw–Hill, New York.

Oard, D. W. and Marchionini, G. (1996). A Conceptual Framework for Text Filtering. Technical Report CS-TR-3643, University of Maryland.

Popovic, M. and Willett, P. (1992). The Effectiveness of Stemming for Natural Language Access to Slovene Textual Data. *Journal of the American Society for Information Science*, 43(5):384–390.

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

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientific Computing, 2nd ed.* Cambridge University Press, Cambridge, UK.

Ragas, H. and Koster, C. H. A. (1998). Four Text Classification Algorithms Compared on a Dutch Corpus. In Croft, W. B., Moffat, A., van Rijsbergen, C. J., Wilkinson, R., and Zobel, J., editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 369–370, Melbourne, Australia. ACM Press, New York.

Richardson, R. and Smeaton, A. F. (1995). Using wordnet in a knowledge-based approach to information retrieval. In *Proceedings of the 17th BCS-IRSG Colloquium on Information Retrieval*, Crewe.

Robertson, S. and Hull, D. A. (2000). The TREC-9 Filtering Track Final Report. In *The Ninth Text REtrieval Conference (TREC-9).*

Robertson, S. and Walker, S. (2000). Threshold Setting in Adaptive Filtering. *Journal of Documentation*, 56:312–331.

Robertson, S. E. (1977). The Probability Ranking Principle. *Journal of Documentation*, 33(4):294–304.

Rocchio, J. J. (1971). Relevance Feedback in Information Retrieval. In *The SMART Retrieval System — Experiments in Automatic Document Processing*, pages 313–323, Englewood Cliffs, NJ. Prentice Hall, Inc.

Rüger, S. M. (1998). Feature Reduction for Information Retrieval. In Voorhees, E. M. and Harman, D. K., editors, *The Seventh Text REtrieval Conference (TREC-7)*, pages 409–412, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-242.

Salton, G. (1975). A Vector Space Model for Information Retrieval. *Communications of the ACM*, 18(11):613–620.

Salton, G. and McGill, M. (1983). *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, New York.

Sanderson, M. (1994). Word Sense Disambiguation and Information Retrieval. In Croft, W. B. and van Rijsbergen, C. J., editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 142–151, Dublin, Ireland. ACM Press.

Schapire, R. E., Singer, Y., and Singhal, A. (1998). Boosting and Rocchio Applied to Text Filtering. In Croft, W. B., Moffat, A., van Rijsbergen, C. J., Wilkinson, R., and Zobel, J., editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 215–223, Melbourne, Australia. ACM Press, New York.

Schölkopf, B. (1998). Support vector machines — a practical consequence of learning theory. *IEEE Intelligent Systems*, 13(4):18–21.

Simons, J., Arampatzis, A., Wondergem, B. C., Schomaker, L. R., van der Weide, T. P., and Koster, C. (2000). Profile — A Multi-Disciplinary Approach to Information Discovery. In *Proceedings of the Second International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2000).*

Singhal, A. (1997). AT&T at TREC-6. In Voorhees, E. M. and Harman, D. K., editors, *The Sixth Text REtrieval Conference (TREC-6)*, pages 215–225, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-240.

Singhal, A., Buckley, C., and Mitra, M. (1997). Learning Routing Queries in a Query Zone. In Belkin, N., Narasimhalu, D., and Willett, P., editors, *Proceedings of the 20st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25–32. ACM Press, New York.

Smeaton, A. and Quigley, I. (1996). Experiments on using semantic distances between words in image caption retrieval. In Frei, H., Harman, D., Schäuble, P., and Wilkinson, R., editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 174–180. ACM Press.

Smeaton, A. F. (1997). Using NLP or NLP Resources for Information Retrieval Tasks. In Strzalkowski, T., editor, *Natural Language Information Retrieval*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Smeaton, A. F., Kelledy, F., and O'Donnell, R. (1995). TREC-4 Experiments at Dublin City University: Thresholding Posting Lists, Query Expansion with WordNet and POS Tagging of Spanish. In Harman, D. K., editor, *The Fourth Text REtrieval Conference (TREC-4)*, pages 373–390, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-236.

Smeaton, A. F., O'Donnell, R., and Kelledy, F. (1994). Indexing Structures Derived from Syntax in TREC-3: System Description. In Harman, D. K., editor, *The Third Text REtrieval Conference (TREC-3)*, pages 55–67, Gaithersburg, Md. 20899. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-225.

Spitters, M. and Kraaij, W. (2000). A Language Modeling Approach to Tracking News Events. In *Notebook of the Topic Detection and Tracking 2000 Workshop*, Gaithersburg, MD, USA. NIST.

Strzalkowski, T. and Carballo, J. P. (1995). Natural Language Information Retrieval: TREC-4 Report. In Harman, D. K., editor, *The Fourth Text REtrieval Conference (TREC-4)*, pages 245–258, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-236.

Strzalkowski, T., Lin, F., and Carballo, J. P. (1997). Natural Language Information Retrieval: TREC-6 Report. In Voorhees, E. M. and Harman, D. K., editors, *The Sixth Text REtrieval Conference (TREC-6)*, pages 347–366, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-240.

Strzalkowski, T., Lin, F., Wang, J., and Perez-Carballo, J. (1999). Evaluating Natural Language Processing Techniques in Information Retrieval — A TREC Perspective. In Strzalkowski, T., editor, *Natural Language Information Retrieval*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

van Hameren, A. (2001). *Loaded Dice in Monte Carlo*. PhD thesis, University of Nijmegen.
`http://arXiv.org/abs/hep-ph/0101094`.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths, London, United Kingdom.

Voorhees, E. and Harman, D. (1998). Overview of the Seventh Text REtrieval Conference (TREC-7). In Voorhees, E. M. and Harman, D. K., editors, *The Seventh Text REtrieval Conference (TREC-7)*, pages 1–24, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-242.

Voorhees, E. and Harman, D. (1999). Overview of the Eighth Text REtrieval Conference (TREC-8). In Voorhees, E. M. and Harman, D. K., editors, *The Eighth Text REtrieval Conference (TREC-8)*, pages 1–24, Gaithersburg, Maryland. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-246.

Voorhees, E. M. (1994). Query Expansion using Lexical-Semantic Relations. In Croft, W. B. and van Rijsbergen, C. J., editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 61–69, Dublin, Ireland. ACM Press.

Voorhees, E. M. and Harman, D. K., editors (1997). *The Sixth Text REtrieval Conference (TREC-6)*. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-240, Gaithersburg, Maryland.

Williams, R. N. (1991). *Adaptive Data Compression*. Kluwer Academic Publishers.

Winograd, W. (1983). *Language as a Cognitive Process*. Addison-Wesley Pub. Co., Reading MA, USA.

Yang, Y. and Pedersen, J. (1997). A Comparative Study on Feature Selection in Text Categorization. In Engels, R., Evans, B., Herrmann, J., and Verdenius, F., editors, *Proceedings of the Fourteenth International Conference on Machine Learning '97 (ICML 97)*, Vanderbilt University, Nashville, TN.

Yang, Y., Pierce, T., and Carbonell, J. (1998). A Study on Retrospective and On-line Event Detection. In Croft, W. B., Moffat, A., van Rijsbergen, C. J., Wilkinson, R., and Zobel, J., editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 28–36, Melbourne, Australia. ACM Press, New York.

Zhai, C. (1997). Fast Statistical Parsing of Noun Phrases for Document Indexing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, DC.

Zhai, C., Tong, X., Frayling, N. M., and Evans, D. A. (1996). Evaluation of Syntactic Phrase Indexing — CLARIT NLP Track Report. In Harman, D. K. and Voorhees, E. M., editors, *The Fifth Text REtrieval Conference (TREC-5)*, pages 347–358, Gaithersburg, Md. 20899. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 500-238.

Index

Samenvatting

In deze dissertatie worden een aantal nieuwe ideeën geïntroduceerd aangaande het opzoeken van digitale informatie (Information Retrieval). In het bijzonder worden aan twee belangrijke onderwerpen aandacht besteed:

- Het filteren van informatie – het selecteren van relevante documenten met betrekking tot een specifiek kennisgebied uit een dynamische informatie omgeving.
- Het representeren van tekstuele informatie – manieren waarop de informatie bevat in een document gerepresenteerd kan worden zodat zoekmethoden er doeltreffend gebruik van kunnen maken.

In hoofdstuk 2 beschrijven wij filtering als een adaptief en tijdsgebonden proces. Een proces dat, in tegenstelling tot het traditionele doorzoeken van teksten, zowel het dynamische gedrag als de tijdsgebonden aspecten van die informatie beschouwt. Dit hoofdstuk beschrijft de theoretische resultaten van experimenteel onderzoek verricht over de laatste jaren. Geleid door de resultaten van deze experimenten formuleren we de vereisten voor effectiviteit en efficiëntie van filtering processen. Dit geeft een coherent beeld op filtering, echter de geformuleerde ideeën laten zich ook toepassen op andere informatie analyse gebieden waarin tijdsgebonden data een rol spelen. In hoofdstuk 3 onderzoeken we het gebruik van distributies van data over de tijd. We gebruiken hierin de hypothese dat data die uniform over de tijd verspreid voorkomen de toekomst beter voorspellen, en daarom meer waardevol zijn. Dit idee is getest gebruik makend van een nieuwe methode voor term selectie genaamd *term occurence uniformity*. In hoofstuk 4 introduceren wij de *score-distributional (S-D) threshold optimization*. Hierin wordt aan de hand van drempels een beslissing genomen om een document te selecteren of niet. Alle beschreven ideeën en modellen zijn geïmplementeerd in het prototype FilterIt systeem gepresenteerd in hoofdstuk 5. Dit systeem heeft zijn waarde, en de haalbaarheid van de daarin geïmplementeerde ideeën, bewezen in de *TREC-9 Filtering Track*.

In IR is de meest eenvoudige representatie van een document de verzameling woorden die het document bevat. In hoofdstuk 6 behandelen wij *linguistically motivated indexing (LMI)*, een alternatieve representatie van documenten waarin de structuur van de taal waarin deze geschreven zijn gebruikt wordt. De voorgestelde LMI gaat om met taal op een coherente en compacte manier zonder een diepgaande syntactische analyse. In hoofdstuk 7 beschrijven wij experimenteel werk met linguïstische bronnen en methoden. Bovendien evalueren wij hier een deel van het voorgestelde LMI.