INTER-DOCUMENT SIMILARITIES, LANGUAGE MODELS, AND AD HOC INFORMATION RETRIEVAL

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Oren Kurland

August 2006

© 2006 Oren Kurland
ALL RIGHTS RESERVED

INTER-DOCUMENT SIMILARITIES, LANGUAGE MODELS, AND AD HOC INFORMATION RETRIEVAL

Oren Kurland, Ph.D.
Cornell University 2006

Search engines have become a crucial tool for finding information in repositories containing large amounts of textual data in unstructured form (e.g., the Web). However, the task of ad hoc information retrieval, that is, finding documents within a corpus that are relevant to an information need specified using a query, remains a hard challenge.

The language modeling approach to information retrieval provides an effective framework for approaching various problems and has yielded impressive empirical performance. However, most previous work on language models for information retrieval focuses on document-specific characteristics to estimate documents' language models, and therefore does not take into account the structure of the surrounding corpus, a potentially rich source of additional information.

We present a novel perspective for approaching the task of ad hoc retrieval: information provided by document-based language models can be enhanced by the incorporation of information drawn from clusters of similar documents that are created offline. We present several retrieval algorithms that are natural instantiations of this idea and that post performance that is substantially better than that of the standard language modeling approach. We also show that the best performing of these algorithms posts state-of-the-art performance for *structural re-ranking*, wherein an initially retrieved subset of the documents is re-ranked to obtain high precision specifically among the first few documents, using inter-document similarities within the list as an extra information source.

As further exploration of the re-ranking approach just described, and inspired by the PageRank and HITS (hubs and authorities) algorithms for Web search, we propose a graph-based framework that applies to document collections lacking hyperlink information. Specifically, *centrality* induced over graphs wherein links represent asymmetric language-model-based inter-document similarities constitutes the basis of effective re-ranking algorithms. Combining our two paradigms for similarity representation — i.e., clusters of documents and links representing language-model-based inter-item similarities — helps to improve the effectiveness of centrality-based approaches. For example, document "authoritativeness" as induced by the HITS algorithm over cluster-document graphs is a highly effective re-ranking criterion. Furthermore, "authoritative" clusters are shown to contain a high percentage of relevant documents.

BIOGRAPHICAL SKETCH

Oren Kurland holds a B.Sc. in Mathematics and Computer Science and an M.B.A. from Tel Aviv University in Israel. In August 2002, Oren joined the Ph.D. program in the Department of Computer Science at Cornell University.

To Ehud, Ilana and Sharon

In memory of Shlomo and Gita Kurland and Izack and Haya Diukman

ACKNOWLEDGEMENTS

I had the good fortune to have Lillian Lee as my advisor. Her thoughtful guidance helped shape me as a researcher. As both a scientist and an advisor, Lillian has been a wonderful role model from whom I draw inspiration. I will always be thankful for that.

I thank my committee members, Shimon Edelman, Johannes Gehrke, and Thorsten Joachims for helpful comments on the work presented in this thesis. I am also thankful to the following people for comments and discussions on parts of the work on which this thesis is based: James Allan, Eric Breck, Jamie Callan, Claire Cardie, Yejin Choi, William Cohen, Bruce Croft, Oren Etzioni, Jon Kleinberg, Art Munson, Fernando Pereira, Filip Radlinski, Ves Stoyanov, and Justin Wick.

I thank ChengXiang Zhai and Victor Lavrenko for answering questions about their work, and Andrés Corrada-Emmanuel and David Fisher for technical assistance with the Lemur toolkit. I also thank the anonymous reviewers of Kurland and Lee (2004), Kurland and Lee (2005), Kurland et al. (2005) and Kurland and Lee (2006) for valuable comments on work that appeared in these papers and that serves as the basis of many chapters of this thesis. In addition, I thank the anonymous reviewers of the ACM Transactions on Information Systems (TOIS) journal for comments on an extended version of Kurland and Lee (2004) and the anonymous reviewers of the Journal of Artificial Intelligence Research (JAIR) for comments on an extended version of Kurland and Lee (2005).

I also want to thank Rimon Barr, Siggi Cherem, Kevyn Collins-Thompson, Carmel Domshlak, Leonid Kontorovitch, Ashwin Machanavajjhala, Bo Pang, Liviu Popsecu, Cindy Robinson and Vivek Vishnumurthy — people, who in their own way, have had a considerable impact on my professional life during the last few years.

I also thank the Computer Science Department at Carnegie Mellon University for its hospitality during the 2004-2005 academic year.

This thesis is based upon work supported in part by the National Science Foundation (NSF) under grants ITR/IM IIS-0081334, IIS-0329064 and CCR-0122581; SRI International under subcontract no. 03-000211 on their project funded by the Department of the Interior's National Business Center; and by an Alfred P. Sloan Research Fellowship. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of any sponsoring institutions, the U.S. government, or any other entity.

Last but not least, I thank my parents, Ehud and Ilana, and my sister, Sharon, for their constant support before and during my years at Cornell.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

One of the major goals for the field of artificial intelligence is to support people's decision making processes. To that end, the ability to obtain information relevant to the task at hand is of utmost importance. The abundance of information (in digital form) available in on-line repositories can be highly beneficial for both humans and automated computer systems that seek information, yet poses extremely difficult challenges due to the variety and amount of data available.

It is therefore not a surprise that in the Web setting — perhaps the best example of an abundance of accessible information — *search engines* have become a crucial tool upon which millions of users are dependent for finding desired information.

One of the core problems that search engines face in order to satisfy users' information needs is "judging" whether a piece of (textual) information is relevant to a given information need as specified by a text query. This research problem has attracted attention for almost forty years by now (Salton, 1968), and is still far from being completely solved.

Indeed, there are various challenges involved in estimating the relevance of a text span to an information need underlying a query. For example, users often use queries that contain very few terms to describe their information needs (Spink and Jensen, 2004), and such queries are in many cases ambiguous. Furthermore, resolving ambiguity of terms in documents might itself be important for handling short queries (Krovetz and Croft, 1992; Sanderson, 1994). Another example of a challenge underlying the task of relevance estimation, just mentioned, is the *word mismatch* problem (Xu and Croft, 1999): relevant pieces of text might not contain (some of) the query terms, but might still discuss the same topic as that of the query. For example, the text span "freight truck" is potentially relevant to the query "shipment vehicles", although there are no terms appearing in both of them.

The work in this thesis establishes new approaches and modeling techniques for approaching the problem of "judging" the relevance of a text document to a text query — the basis of the *ad hoc information retrieval* task:

> Given a textual query comprised of several terms and a static repository (*corpus*) of documents, rank the documents in the repository by their estimated relevance to the query.

There is a long history of research on devising effective retrieval algorithms that can find (many) relevant documents in response to a query and position them at high ranks within the returned list of results. Books such as van Rijsbergen (1979), Grossman and Frieder (1998) and Baeza-Yates and Ribeiro-Neto (1999) discuss various retrieval models and provide many references to pertinent literature.

Among the well known paradigms for ad hoc retrieval are the vector-space model (Salton, 1968; Salton et al., 1975), wherein both the query and the documents are represented as vectors in a vector space and ranking is based on similarity in this space; probabilistic approaches (Maron and Kuhns, 1960; Robertson and Sparck Jones, 1976; Croft and Harper, 1979; Sparck Jones et al., 2000) that estimate the probability that a document is relevant to a query; the inference network model (Turtle and Croft, 1990), which is based on a Bayesian network model, and logic-based approaches (van Rijsbergen, 1986).

In 1998, Ponte and Croft (1998) proposed a new retrieval paradigm based on (statistical) *language models* (LMs). (See Rosenfeld (2000) for survey of language models). The basic retrieval method proposed in Ponte and Croft's paper is to rank documents in a corpus by the probability that their induced language models generate the query terms, where a language model is a probability distribution defined over a fixed vocabulary. Among the advantages stated by Ponte and Croft in using LMs for ad hoc retrieval are the ability to exploit models developed in other realms in which language models have been used (e.g., speech recognition) and impressive empirical performance (Ponte and Croft, 1998). Indeed, Ponte and Croft's paper ignited a new line of research in information retrieval utilizing language models. (In Section 2 we survey work on using language models for ad hoc information retrieval, and present some of the connections of the language-model approach to previously proposed retrieval models.)

However, most of the existing work on utilizing language models for ad hoc retrieval has focused on document-specific characteristics for estimating document language models and ignored a potentially rich source of helpful information — the *corpus structure*. In this thesis we show that incorporating corpus-structure information — modeled using clusters of similar documents — into the language modeling framework for information retrieval can result in highly effective retrieval algorithms that substantially outperform the standard language modeling approach.

In a conceptually similar vein, we show that modeling the structure of an initially retrieved (short) list of documents as manifested in inter-document similarities within the list can lead to the design of highly effective *re-ranking* algorithms; such algorithms re-order the documents in the list to obtain high precision at top ranks. Specifically, we present two approaches for this re-ranking setting: one adapts the algorithms mentioned above (originally designed for ranking all documents in a corpus), and the other utilizes a graph representation of inter-item similarities. Language models play an important role in both approaches — they serve as a means for inducing similarity between spans of texts.

Thus, the work presented in this thesis encompasses several novel contributions for approaching the ad hoc retrieval task. The first is a novel perspective: information provided by document-based language models can be enhanced by the incorporation of corpus-structure information manifested in clusters of similar documents. We make a distinction between two roles that clusters can play: selecting relevant documents and smoothing document language models, and show that the combination of the two yields very effective retrieval algorithms. Furthermore, we show that overlap of clusters is highly important for effectively exploiting corpus structure using our retrieval algorithms. Additionally, we show that our best performing algorithm yields state-of-the-art performance not only for ranking all documents in a corpus, but also for re-ranking an initially retrieved (short) list, using clusters created from documents in this list — i.e., *query-dependent clusters*.

Another contribution is a graph-based framework for the re-ranking setting just mentioned, which is based on *centrality* induction over graphs wherein links are induced by asymmetric language-model-based inter-document similarities. We show that centrality of documents in the initial list and relevance are connected, and that using centrality as a document "bias" in the language-model framework for retrieval results in highly effective re-ranking algorithms.

Combining our two paradigms for similarity representation — i.e., clusters of similar documents and links representing language-model-based inter-item similarities — helps to improve the effectiveness of centrality-based approaches for re-ranking. For example, document "authoritativeness" as induced by the HITS (hubs and authorities) algorithm (Kleinberg, 1998) over cluster-document graphs is a highly effective re-ranking criterion. Furthermore, "authoritative" clusters are shown to contain a high percentage of relevant documents.

1.1 Thesis outline

We begin in Chapter 2 by reviewing previous work on utilizing language models for ad hoc retrieval and introducing notation that is used throughout the thesis. We conclude Chapter 2 with the observation that for deriving document language models within the language-model framework for ad hoc retrieval, the focus of most existing work has been on document-specific characteristics rather than on combining this information with information about the "context" of the documents with respect to the corpus in which they reside. We then proceed to Chapter 3 wherein we survey past work in ad hoc retrieval that utilizes corpus-structure-information induced in a query-independent fashion.

In Chapter 4 we present an algorithmic framework for ad hoc retrieval that utilizes language models built from clusters created offline. Several retrieval algorithms that are instantiations of this framework are shown to be highly effective. Our best performing algorithm posts significant improvements over the basic LM approach and is competitive with highly optimized state-of-the-art pseudo-feedback-based methods.

While in Chapter 4 our retrieval algorithms rank all documents in a corpus in response to a query, in Chapter 5 we focus on a *structural re-ranking* approach to the ad hoc retrieval task: an initially retrieved (short) list of documents is re-ranked to obtain high precision at top ranks, using inter-document similarities within the list. Adaptations of algorithms from Chapter 4 to this setting yield very good performance results.

We conclude Chapter 5 by showing that pairwise inter-document similarities constitute a useful source of information for re-ranking an initial list. We further pursue this idea in Chapter 6, in which we present a graph-based framework for structural re-ranking, wherein language-model-based asymmetric inter-document similarities are used to induce links between documents. Retrieval algorithms based on *centrality* induction over our constructed graphs — using graph-based methods adopted from Web retrieval such as PageRank (Brin and Page, 1998) — post substantial improvements over the initial ranking of the list upon which re-ranking is performed.

In Chapter 7 we incorporate cluster-based information into the graph-based framework from Chapter 6, which utilizes document-only graphs. We show that document *authority* scores as induced by the HITS algorithm (Kleinberg, 1998) over cluster-document bipartite graphs constitute a highly effective re-ranking criterion. Furthermore, "authoritative" clusters are shown to contain a high percentage of relevant documents.

In Chapter 8 we compare the performance of our various structural re-ranking algorithms (from Chapters 5, 6 and 7), and show that two of our best performing algorithms outperform state-of-the-art pseudo-feedback models.

Finally, we summarize our contributions and present future directions in Chapter 9.

1.2 Evaluation methodology (design choices)

Throughout this thesis we use TREC data (e.g., Voorhees and Harman (2000)) to evaluate our algorithms' performance, since this data provides document collections, queries and (partial) relevance judgments, as is standard in information retrieval research. However, there are some (arguable) choices that we have made with regard to evaluation that are worth commenting on in some detail.

Measures While there is a wide variety of evaluation measures for estimating the effectiveness of retrieval algorithms, one should choose those pertaining to the specific goal of the retrieval system (Buckley and Voorhees, 2000). In Chapter 4, wherein our algorithms rank all documents in a corpus in response to a query, we use *MAP* (mean average non-interpolated precision) at 1000 to evaluate the ranking quality of the different algorithms (Harman and Voorhees, 1998). This standard TREC-evaluation metric measures both the percentage of relevant documents (with respect to the total number of relevant documents) that are among the 1000 highest-ranked retrieved results and their positioning. Specifically, to calculate average non-interpolated precision for a single query, we traverse the list of retrieved documents (from the highest ranked document to the document at rank 1000), and average the precision at ranks at which relevant documents are located with respect to the total number of relevant documents; the mean of these single-query values is the MAP value (Harman and Voorhees, 1998). Although MAP is connected with recall, we also present performance results for the latter (at 1000). We believe that an IR system that is able to "collect" many of the relevant documents in the corpus within a relatively short list (with respect to the number of documents in the ambient corpus) can be of a great value in interactive retrieval settings, for example. In such settings, effective tools for visualization of retrieved results can help users to quickly detect relevant documents in the retrieved list (Hearst and Pedersen, 1996; Leuski, 2001). Furthermore, systems capable of obtaining high recall can serve as a first step of a retrieval process wherein the second step is to (automatically) re-rank the initial list to obtain high precision at top ranks. The re-ranking algorithms presented in Chapters 5, 6 and 7 are designed to attain this goal.

Chapters 5, 6 and 7 focus on the re-ranking approach, as just mentioned, and therefore the natural evaluation measures to use in them are those evaluating the precision of the very few most highly-ranked returned results. Therefore, we use the precision at the top 5 and 10 documents (henceforth prec@5 and prec@10, respectively) and the mean reciprocal rank (MRR) of the first relevant document (Shah and Croft, 2004) as evaluation measures in these chapters.

Parameter tuning The retrieval algorithms presented throughout this thesis (whether ours or the reference comparisons we use) incorporate some free parameters. The performance numbers correspond to choosing parameter values that result in optimized performance with respect to a *single* evaluation measure. (We discuss below what data this optimization is performed on.) This approach represents our belief that performance numbers presented for an algorithm should represent a *single instance* of parameter values rather than multiple instantiations (e.g., presenting prec@5 and prec@10 results for two different parameter settings, each chosen to optimize one of the two measures). We believe that such an approach provides a more realistic picture of the expected performance patterns of an algorithm. The latter observation is obvious in a setting wherein one employs two "competing" measures (e.g., recall and absolute precision), since optimizing performance for one of the two measures can result in very low performance numbers for the other measure; however, we found that the observation also holds if two seemingly correlated measures are employed (e.g., prec@5 and prec@10).

To choose the evaluation measure for which performance is optimized, we consider the main criterion by which we would like to evaluate the retrieval algorithm. In Chapter 4, wherein our algorithms rank all documents in the corpus, we are interested in the general quality of the ranking induced by the different methods and thus the natural evaluation measure to optimize performance for is MAP (Buckley and Voorhees, 2000). On the other hand, in Chapters 5, 6 and 7 we use prec@5 as the evaluation measure for which performance is optimized, as we are interested in the precision at top ranks of the list; alternatively, prec@10 can be used. (See Harman and Voorhees (1998) for a discussion of potential issues with MRR.)

To optimize parameter values, one can use a training set of documents with a set of queries for which relevance judgments are available, and then use a different set of queries (perhaps with a different set of documents) for testing the retrieval performance (Joachims, 2002; Nallapati, 2004; Metzler, 2005; Metzler and Croft, 2005).(For examples in language-model-based systems see e.g., Gao et al. (2004)

and Liu and Croft (2004).) We, however, optimize parameter values for the same queries we present results for. (Refer to Mitra et al. (1998), Lafferty and Zhai (2001) and Zhai and Lafferty (2001a) for examples of this parameter tuning approach. The latter two focus on language-model-based retrieval approaches.).

Thus, we focus on the merits of incorporating inter-document similarity information into the retrieval framework rather than on exploring the question of whether the parameter values estimated using one corpus and set of queries can be effective for other choices of queries/corpora as well. Indeed, the question of whether the retrieval performance patterns observed in one benchmark are expected to generalize to other benchmarks is a subject of ongoing inquiry in the information retrieval community (Zobel, 1998; Buckley and Voorhees, 2000; Voorhees, 2000; Sanderson and Zobel, 2005). We hasten to point out, however, that we test our algorithms with a variety of corpora and queries to ensure that the merits of our methods are not specific to a certain choice of corpus and/or set of queries (Buckley and Voorhees, 2000). Furthermore, the performance results we present are averages over sets of queries, and thus the single parameter-values instance for which performance is presented represents an optimal choice with respect to the average performance over the set of tested queries rather than single queries. In addition, to examine whether the performance numbers presented indeed represent consistent improvements (or lack thereof) over the chosen baselines for the entire set of queries at hand, and not improvements over very few queries (Zobel, 1998; Sanderson and Zobel, 2005), we present the statistical significance (or lack thereof) of the performance differences between our algorithms' performance and that of the tested baselines.

Aside: On (not) generalizing over queries. The question of whether parameter values estimated using a held-out set (of documents/queries) can be useful for other benchmarks is an important research question by itself — although not tackled in this thesis — and depends on various factors, such as the characteristics of the document collection, query characteristics, and more.

The variability of query characteristics (and the characteristics of the information needs they represent), for example, is a highly important factor affecting the performance patterns of a retrieval technique that employs the same parameter values for different queries, even if the queries are run with the same corpus. In the language-modeling approach, for example, query length has a considerable impact on the desired *smoothing* approach employed to estimate document language models (Zhai and Lafferty, 2001b). (See Section 2 for details.) Furthermore, intrinsic query characteristics (e.g., level of ambiguity, vocabulary (or word form) mismatch between the query and documents in the corpus) might affect the inter (and intra) class structure of relevant and non-relevant documents. For example, assuming a vector-space representation for documents (Salton et al., 1975), while for one query the classes of relevant and non-relevant documents might be "linearly separable" (i.e., there is an hyperplane separating the space into two sub-spaces, each containing documents from only one class), for other queries this might not be the case and relevant and non-relevant documents might form tight clusters that are scattered in different portions of the space. Thus, completely different parameter values might be needed to appropriately tackle these differences if one applies the same retrieval approach for all queries; perhaps even different retrieval approaches are called for to handle such cases.

Recent research in ad hoc retrieval addresses the issue of query-characteristic variability just mentioned (Cronen-Townsend et al., 2002, 2004; Yom-Tov et al., 2005), and evaluation paradigms have also been designed to tackle related issues — e.g., the *Robust Retrieval Track* of TREC (Voorhees, 2005). We believe that this constitutes an important direction towards designing retrieval systems that identify some query characteristics (with respect to a given document collection) and classify the queries into some pre-defined classes of queries with similar characteristics, along the spirit of recent suggestions (Voorhees, 2005) and ideas employed for *question answering* (Voorhees, 2002); such retrieval systems, then, could automatically choose methods and parameter values appropriate for the specific class of each query, as learned before hand[1]. In this spirit, Cronen-Townsend et al. (2004), for example, try to predict what queries will benefit from *query expansion* and what queries should be used directly without any expansion. However, designing mechanisms for analyzing the "class" of a specific query, along with choosing appropriate retrieval approaches and parameter values given the presumed class, is out of the scope of this thesis.

[1] We point out that a different conceptual approach, which is based on devising retrieval functions that rise from a proposed *axiomatic* system, has recently been shown to help to ameliorate the problem of performance robustness with respect to parameter values (Fang and Zhai, 2005).

Chapter 2
The language modeling approach to information retrieval

In this chapter we present the concept of *statistical language models* which has been extensively used in various applications such as machine translation, text categorization and character recognition. We then provide an overview of the development of the language modeling (LM) approach to ad hoc retrieval; the overview is concluded with the observation that most work on inducing document language models has not exploited a potentially rich source of information — *corpus structure*. In Section 2.3 we present the language-model-based estimates used throughout this thesis for inducing similarities between spans of text, taking care of length-bias and numerical problems.

2.1 Language models

The term (statistical) "language model" (LM) refers to a function defined over sequences of terms drawn from a fixed vocabulary, utilizing (occurrence) statistics associated with the sequences. (See Rosenfeld (2000) for a survey of language models). Usually, the function values represent probabilities derived using some modeling assumptions. By now, statistical language models have been used for more than twenty years in numerous applications such as speech recognition, machine translation, character recognition, text classification, and more (Rosenfeld, 2000).

A commonly used language model is the *n-gram language model*, which is based on the assumption that the probability of observing term t_i, which appears at position i of the text, depends (only) on the $n-1$ preceding terms $t_{i-n+1}, \ldots, t_{i-1}$. A *bigram* language model, for example, is an n-gram language model with $n = 2$, i.e., the probability of observing a term at a given position in the text depends only on the preceding term. Setting $n = 1$ we get the *unigram language model*, which encodes a term independence assumption (a.k.a. *bag of words* assumption).

Estimation of n-gram language model probabilities from text is based on occurrence frequencies. Specifically, we can estimate the language model probability $p(t_i|t_{i-n+1}, \ldots, t_{i-1})$ from text span x using the so-called *maximum likelihood estimate* (MLE):

$$\widetilde{p}_x^{MLE}(t_i|t_{i-n+1}, \ldots, t_{i-1}) \stackrel{def}{=} \frac{tf\,(t_{i-n+1}, \ldots, t_i \in x)}{tf\,(t_{i-n+1}, \ldots, t_{i-1} \in x)}, \tag{2.1}$$

where $tf(y \in x)$ denotes the number of times the term sequence y occurs in x. A common practice for extending the term-based distribution $\widetilde{p}_x^{MLE}(t_i|t_{i-n+1}, \ldots, t_{i-1})$ to a distribution over sequences is to multiply the individual probabilities:

$$p_x^{MLE}(t_1, \ldots, t_m) \stackrel{def}{=} \prod_{i=i}^{m} p(t_i|t_{i-n+1}, \ldots, t_{i-1}). \tag{2.2}$$

(For early positions in the text, we back off to a lower-order n-gram model. That is, for the first term t_1 in the text we use the unigram language model probability $\widetilde{p}_x^{MLE}(t_1) = \frac{tf(t_1 \in x)}{|x|}$, for the second term t_2 we use the bigram language model probability $\widetilde{p}_x^{MLE}(t_2|t_1)$, and so forth.)

Using the maximum-likelihood-based estimate just described (Equation 2.2) for assigning an n-gram language model probability to a text span that contains a term sequence of length n that does not appear in the text used for language model estimation will result in an assigned probability of zero. This is naturally a problem (known as the *zero probability problem*) since the fact that a text used for estimation does not contain a certain term sequence does not imply that the probability that this sequence occurs in another text is zero; however, the (maximum likelihood) estimates involved will be zero. One approach often employed for handling this problem (also termed the *sparse data problem*) is *smoothing* — assigning non-zero probabilities to term sequences not observed in the text used for language model estimation. In Section 2.3 we present the Dirichlet smoothing technique, which was shown to be quite effective for the ad hoc retrieval task (Zhai and Lafferty, 2001b). A good survey of smoothing techniques for language models can be found in Chen and Goodman (1998).

A detailed description of our language model induction technique and some estimates appears in Section 2.3.

2.2 Using language models for ad hoc information retrieval

Ponte and Croft (1998) were the first to employ language models for the task of ad hoc information retrieval, using the following ranking principle:

> *The query likelihood ranking principle: rank documents by the probability of their language models generating the query terms.*

(When we say that a language model "generates" a term (sequence), we (roughly) mean that the term (sequence) is thought of as a sample from the vocabulary according to the distribution defined by the language model.)

One potential intepretation of this ranking principle is that a query could be viewed as a sample of terms from the language model induced from a relevant document, and documents are ranked by the probability that their language models represent a "relevance" distribution. Sparck Jones et al. (2003), for example, raise an objection with respect to the theoretical validity of the LM-based ranking principle just mentioned, on the grounds that relevance is not modeled explicitly and that the approach implies that there exists a single relevant document. Work on relevance language models (Lavrenko and Croft, 2001) (see details below) addresses this issue.

While the basic language modeling approach and the classic vector-space model (Salton, 1968) seem to represent two completely different paradigms (at first sight at least), as the information representation methods and therefore the document-query similarity notions are quite different, some work (Hiemstra and Kraaij, 1999; Zhai and Lafferty, 2001b) connects the two paradigms by reasoning about the tf.idf weighting scheme, used in vector-space models, from an LM perspective (Hiemstra and Kraaij, 1999), and explaining some of the roles of smoothing in the LM approach using a vector-space perspective (Zhai and Lafferty, 2001b).

To connect the language-modeling approach to *probabilistic retrieval* (see Fuhr (1992) and Cresanti et al. (1998) for surveys of probabilistic approaches to IR, and Sparck Jones et al. (2000) for a more recent study of a probabilistic approach), Lafferty and Zhai (2003) use the work of Sparck Jones et al. (2000) as a representative for probabilistic approaches. They start from the basic question that Sparck Jones et al. post: "What is the probability that this document is relevant to this query?", and develop a mathematical derivation (using Robertson's *probability ranking principle* (Robertson, 1977)) that under different independence assumptions results either in a language-model-based retrieval criterion or in the probabilistic model of Sparck Jones et al. (2000). An important observation raised by Lafferty and Zhai is that the two models (language-model and probabilistic model) have complementary strengths with regard to estimation, due to the different independence assumptions underlying the models. Case in point: in the language-model approach one estimates the "generation" of a (short) query by a (long) document, while in the probabilistic model the "generation" is supposedly in the reverse direction, making the former estimation task "easier" (from a statistics point of view) than the latter, because documents are much longer than queries and therefore provide more data. On the other hand, exploitation of user feedback is much easier in the probabilistic model than in the language-model framework.

Other work on language models for ad hoc IR suggested alternative derivations (and motivation) for using LMs in the ad hoc retrieval setting (see Hiemstra (2001) for elaborated discussion), but the resultant ranking criteria were very similar (and in many cases equivalent) to that of the query likelihood model. The hidden Markov model presented by Miller et al. (1999), for example, assumes that a term in a query is either generated by the (unsmoothed) unigram document language model or by a background language model (i.e., the corpus language model). The resultant relevance scoring method (in its simplest form) is therefore based on the probability that the query is generated by the smoothed document language model, where smoothing is essentially a linear interpolation of the document and background language models.

The *risk minimization framework* of Lafferty and Zhai (2001) associates a "loss" with the act of selecting a document in response to a query. Such a loss can be measured in several ways, one of which is by the "difference" (distance) between the query language model and the document language model. One way to estimate this difference is by measuring the *Kullback-Leibler (KL)* divergence (Cover and Thomas, 1991) between the document and query language models; the resultant retrieval paradigm is therefore known in the literature as the *KL divergence retrieval framework*. An important observation made by Lafferty and Zhai (2001) is that for ranking purposes this approach is equivalent to the query likelihood model if unigram language models are used. We further discuss this equivalence in Section 2.3.

The sparse data problem mentioned above — a challenge for the estimation of document language models — has drawn quite a lot of attention and several smoothing methods have been studied in the IR literature (Song and Croft, 1999; Zhai and Lafferty, 2001b, 2002; Zaragoza et al., 2003), most of which exploit corpus statistics regarding term occurrences. Furthermore, smoothing in the language-model framework was used as a means for explaining the importance of terms and their weights in different scoring functions (Zhai and Lafferty, 2001b; Hiemstra, 2002).

While most work on language models for ad hoc IR utilizes unigram language models, several researchers explored the use of language models that capture higher order dependencies between terms (Song and Croft, 1999; Nallapati and Allan, 2002; Gao et al., 2004; Srikanth and Srihari, 2004; Gao et al., 2005). Some of these models were shown to post moderate improvements over a unigram-language-model-based approach in terms of resultant retrieval performance. It is important to observe here that while one could expect improvements when moving away from the (unrealistic) term-independence assumption, it is not, however, the case that a "better" language model — in terms of predicting term sequences in a new text — necessarily results in a better retrieval performance (Azzopardi et al., 2003; Morgan et al., 2004). Indeed, recent work suggests that the objective function for tuning language-model (or more generally retrieval-function) parameters should be the evaluation metric used to measure retrieval performance (Morgan et al., 2004; Metzler, 2005; Metzler and Croft, 2005), and not the predictive power of the language model.

Another challenge addressed by work on language models (and other retrieval models for this matter) for ad hoc retrieval is the *word mismatch* problem (Xu and Croft, 1996): relevant documents might contain terms that discuss the same topic as that of the query, but might not necessarily contain the query terms themselves. One solution employed by language-model approaches (and retrieval approaches in general) is *query expansion*: "augmenting" the original unsmoothed query language model with terms considered to be related to the query terms. Query expansion could be viewed as helping with both the *synonomy* problem — two different terms having the same meaning, and the *polysemy* problem — one term having two different meanings.

Very few methods that model general corpus-based term relationships have been explored in the language-model framework (Berger and Lafferty, 1999; Lafferty and Zhai, 2001). On the other hand, *pseudo-feedback*-based methods (e.g., (Ponte, 1998; Lavrenko and Croft, 2001; Lafferty and Zhai, 2001; Zhai and Lafferty, 2001a; Tao and Zhai, 2004)), that is, methods that use the most highly ranked documents from an initial search to define a new "query model", have become the main tool for performing query expansion. The basic concept underlying pseudo-feedback-based approaches to query expansion in the language modeling framework is that terms that are assigned high probability by (many) language models induced from the top retrieved documents with respect to the initial search should be used to augment the original (unsmoothed) query language model. Moreover, such terms should be assigned high probabilities by the expanded model. Under some models, the newly defined language model assigns probabilities to the original query terms that are different than those assigned by the original query language model (Zhai and Lafferty, 2001a).

Another stream of research that led to models that could be considered (in implementation) as query expansion techniques is the work on *relevance models* (Lavrenko and Croft, 2001, 2003; Lavrenko, 2004). The assumption is that there exists a language model — termed relevance model — that generates terms both in the query and in the relevant documents. The different methods for estimating such a relevance model are usually based on a pseudo-feedback approach, and the resultant language model is then used for ranking documents (Lavrenko and Croft, 2003). A (conceptually) similar approach (though different in implementation) that also uses the set of most highly ranked documents to define a "relevance" probability distribution over all terms in a vocabulary is presented by Zhai and Lafferty (2001a). In a similar conceptual vein, following Sparck Jones et al. (2003), Hiemstra et al. (2004) use mixture models to define *parsimonious* language models that help to model the discriminative nature of terms with respect to the classes of relevant and non-relevant documents.

It is important to note that the above mentioned pseudo-feedback methods could be viewed as exploiting *corpus structure* via a *query-biased sampling* approach. For example, terms that co-occur with query terms in the highest ranked documents with respect to an initial search — i.e., documents that exhibit a high surface similarity to the query — might be considered as related to the query topic, thus helping to cope with the synonomy problem, for example. Furthermore, if many of the highest ranked documents from the initial search "discuss" the same topic, and if this topic is the same as that underlying the query, then query expansion performed using these documents could potentially ameliorate the polysemy problem.

But, one main drawback of most pseudo-feedback-based approaches is sensitivity to the quality of the initial ranking as is often manifested in sensitivity to the number of top retrieved documents utilized (Tao and Zhai, 2004). One obvious explanation for this sensitivity is that not all top retrieved documents from the initial search are relevant to the query — in fact many of them are not — and therefore the language model defined from these documents might be biased by term distributions of non-relevant documents that exhibit high surface similarity to the query. Furthermore, recent analysis (Buckley, 2004; Harman and Buckley, 2004) implies that the highest ranked documents in the initially retrieved list often do not exhibit one of the query aspects. Therefore, a language model defined using these documents might not reflect all query aspects. (We hasten to point out, however, that this drawback is common to many retrieval approaches (Buckley, 2004).)

While there has been a lot of focus in the language modeling framework on "improving" the original query model by using "corpus context" as manifested in the most highly ranked documents with respect to an initial search (as described above), the document language models themselves are induced based on document-specific characteristics, and a potentially rich source of information — the structure of the surrounding corpus as manifested in inter-document similarities — has not been accounted for (e.g., for modeling document context in a query-independent fashion). Such information can help ameliorate the problems of synonomy and polysemy as terms appearing in similar documents could be considered (under some modeling assumptions) somewhat related. Moreover, exploiting this information might also help with the sparse data problem. In Chapter 3 we present an example of a scenario wherein a document does not contain any of the query terms, but can be still retrieved (with high score) if associated with similar documents that contain query terms. In fact, in this particular example a pseudo-feedback method for query expansion is not guaranteed to be effective, because the initially retrieved list of documents exhibits two different aspects (topics), one of which is not related to the query but can still significantly affect the way the original query is expanded.

As a last note for this short overview on the use of language models for ad hoc retrieval, we emphasize that language models have been used for other information retrieval tasks (and in different settings of the ad hoc retrieval task) as well. Examples include information filtering (Zhang et al., 2002), text classification (Yang and Liu, 1999; Teahan and Harper, 2003), topic detection and tracking (Spitters and Kraaij, 2001; Lavrenko et al., 2002a; Kraaij and Spitters, 2003), cross lingual retrieval (Hiemstra and Kraaij, 1999; Lavrenko et al., 2002b), distributed retrieval (Xu and Croft, 1999; Si et al., 2002), entry-page search (in Web retrieval) (Kraaij et al., 2002), and the ad hoc retrieval task with queries that contain operators (Hiemstra, 2001), and documents that are marked with structure information (Ogilvie and Callan, 2003).

2.3 Language-model-based similarity induction: length, entropy and numerical effects[1]

In this section we describe our methods for inducing language models and estimating the probabilities they assign to text sequences — probabilities that serve to estimate the strength of association (similarity) between texts of varying lengths.

For language model estimation we treat queries and documents as term sequences. In this section we use the term cluster to refer to the "big document" that results from concatenating the cluster's constituent documents; the order of

[1] The work presented in this section is based on two papers written with Lillian Lee (Kurland and Lee, 2004, 2005) that appeared in the proceedings of SIGIR 2004 and SIGIR 2005 respectively, and on a paper written with Lillian Lee and Carmel Domshlak (Kurland et al., 2005) that appeared in the proceedings of SIGIR 2005.

concatenation has no effect since we are only going to define unigram language models that embed a term independence assumption. We hasten to point out, though, that our retrieval algorithms, presented throughout the following chapters, do not depend on a term independence assumption. In fact, any language-model-based similarity induction method that copes with the issues we elaborate below could be used for our algorithms. Our choice of utilizing unigram language models mainly stems from simplicity of estimation, and the ability to compare performance results to those of state-of-the-art methods in the language modeling framework that in most of the cases utilize unigram language models.

2.3.1 Estimation details

Let $tf(t \in x)$ denote the number of times the term t occurs in the text or text collection x. What is often called the maximum-likelihood estimate (MLE) of t with respect to x is defined as

$$\widetilde{p}_x^{MLE}(t) \stackrel{def}{=} \frac{tf(t \in x)}{\sum_{t'} tf(t' \in x)}.$$

Prior work in language-model-based retrieval (Zhai and Lafferty, 2001b) advocates the use of a *Dirichlet-smoothed* estimate:

$$\widetilde{p}_x^{[\mu]}(t) \stackrel{def}{=} \frac{tf(t \in x) + \mu \cdot \widetilde{p}_{\mathcal{C}}^{MLE}(t)}{\sum_{t'} tf(t' \in x) + \mu},$$

where $\mathcal{C}$ is the corpus; the smoothing parameter μ controls the degree of reliance on relative frequencies in the corpus rather than on the counts in x. As mentioned above, smoothing helps to avoid the zero probability problem, namely, the assignment of zero probability to unseen terms. (See Zhai and Lafferty (2001b; 2002) for detailed discussion of smoothing language models for information retrieval.)

Both estimates just described are typically extended to distributions over term sequences by assuming that terms are independent: for an n-term text sequence $t_1 t_2 \cdots t_n$,

$$\begin{aligned} p_x^{MLE}(t_1 t_2 \cdots t_n) &\stackrel{def}{=} \prod_{j=1}^{n} \widetilde{p}_x^{MLE}(t_j); \\ p_x^{[\mu]}(t_1 t_2 \cdots t_n) &\stackrel{def}{=} \prod_{j=1}^{n} \widetilde{p}_x^{[\mu]}(t_j). \end{aligned}$$

While for the standard language modeling approach this estimator is used to estimate a probability for a fixed short query, in several of our settings we would like to estimate probabilities for different (and much longer) textual items such as clusters and documents. Now, observe that for both $p_x^{MLE}(\cdot)$ and $p_x^{[\mu]}(\cdot)$, long text sequences tend to be assigned lower probabilities (as the number of multiplicands in the equations above increases, and the value of each is less than one, if the text contains at least two different terms) and this will have detrimental effects on our probability estimates for documents and clusters if the (smoothed) MLE is to be used directly[2]. In addition, the value of $p_x^{[\mu]}(\cdot)$ can cause underflow problems due to the long term sequences that comprise documents and clusters.

As mentioned above, throughout the following chapters we use language model probabilities to estimate the strength of association (similarity) between spans of text of varying lengths. Therefore, to cope with the issues incurred by the direct use of the (smoothed) MLE, we adopt an estimation approach that utilizes the *cross-entropy* measure in the following way. For estimating the strength of association between text x (representing either a document's or a cluster's term sequence) and word sequence $\vec{t} = t_1 t_2 \cdots t_n$ (in our settings, a term sequence that comprises a query, a document or a cluster), we use

$$p_x^{CE,\mu}(\vec{t}) \overset{def}{=} \exp\left(-CE(\widetilde{p}_{\vec{t}}^{MLE}(\cdot), \widetilde{p}_x^{[\mu]}(\cdot))\right), \tag{2.3}$$

where $CE(\cdot,\cdot)$ is the cross-entropy between two term-based distributions.

It is important to note that this estimate does not form a valid probability distribution, because the sum over all possible $\vec{t}$ is not 1. In fact, the basic unigram language model itself does not form a proper probability distribution, although it could be adjusted to be so. We do, however, use the probability notation p for this estimate to denote that the estimate is based on language-model probabilities. (In fact, as will be shown below, this estimate is a length-normalized version of the "probability" assigned by the smoothed language model $\widetilde{p}_x^{[\mu]}(\cdot)$ to the term sequence $\vec{t}$.) Moreover, where required, we can scale the estimate to form a probability distribution.

Now, by the definition of cross entropy we get that

$$p_x^{CE,\mu}(\vec{t}) = \exp\left(\sum_v \widetilde{p}_{\vec{t}}^{MLE}(v) \log\left(\widetilde{p}_x^{[\mu]}(v)\right)\right),$$

where v is a term in the vocabulary, and can occur zero or more times in $\vec{t}$.

Since $\widetilde{p}_{\vec{t}}^{MLE}(\cdot)$ is an unsmoothed MLE, we get that $\widetilde{p}_{\vec{t}}^{MLE}(v) = 0$ for $v \notin \vec{t}$, resulting in

$$p_x^{CE,\mu}(\vec{t}) = \exp\left(\sum_{v \in \vec{t}} \widetilde{p}_{\vec{t}}^{MLE}(v) \log\left(\widetilde{p}_x^{[\mu]}(v)\right)\right).$$

[2] Note that the fact that a language model (induced from some source text) assigns a lower probability to a long text than it assigns to a shorter text is unjustified if the probability serves to model the strength of association (or similarity) between the source text and the text to which the probability is assigned — as is assumed throughout this thesis. However, there are settings wherein the assumption is that long texts are less probable to be generated and therefore the above mentioned length-bias does not constitute a problem.

Now, by the definition of the MLE and using algebraic manipulations we get that:

$$\begin{aligned}
p_x^{CE,\mu}(\vec{t}) &= \exp\left(\sum_{v\in\vec{t}} \frac{tf(v\in\vec{t})}{|\vec{t}|}\log\left(\widetilde{p}_x^{[\mu]}(v)\right)\right)\\
&= \exp\left(\sum_{v\in\vec{t}}\log\left(\widetilde{p}_x^{[\mu]}(v)^{\frac{tf(v\in\vec{t})}{|\vec{t}|}}\right)\right)\\
&= \exp\left(\log\prod_{v\in\vec{t}}\left(\widetilde{p}_x^{[\mu]}(v)^{\frac{tf(v\in\vec{t})}{|\vec{t}|}}\right)\right)\\
&= \prod_{v\in\vec{t}}\left(\widetilde{p}_x^{[\mu]}(v)^{\frac{tf(v\in\vec{t})}{|\vec{t}|}}\right) = \prod_{v\in\vec{t}}\left(\widetilde{p}_x^{[\mu]}(v)^{tf(v\in\vec{t})}\right)^{\frac{1}{|\vec{t}|}}\\
&= \left(\prod_{t_j}\widetilde{p}_x^{[\mu]}(t_j)\right)^{\frac{1}{|\vec{t}|}}\\
&= p_x^{[\mu]}(\vec{t})^{\frac{1}{|\vec{t}|}}.
\end{aligned}$$

The above derivation shows that the estimator from equation 2.3 is the geometric mean of the probability assigned by the smoothed unigram language model induced from x to the text sequence $\vec{t}$:

$$p_x^{CE,\mu}(\vec{t}) = p_x^{[\mu]}(\vec{t})^{\frac{1}{|\vec{t}|}}, \tag{2.4}$$

which was also used in work on topic detection and tracking (Lavrenko et al., 2002a). This proposed estimator compensates for the length penalization incurred by the (smoothed) MLE, and ameliorates the above mentioned underflow problem.

Another estimator that we will use in Chapters 5, 6, 7 and 8 is based on the Kullback Leibler divergence (Cover and Thomas, 1991) (denoted as $D(\cdot||\cdot)$):

$$p_x^{KL,\mu}(\vec{t}) \overset{def}{=} \exp\left(-D(\widetilde{p}_{\vec{t}}^{MLE}(\cdot)||\widetilde{p}_x^{[\mu]}(\cdot))\right), \tag{2.5}$$

Now, by the definition of the KL divergence and cross entropy, for any two probability distributions p and q the following holds

$$D(p||q) = CE(p,q) - H(p),$$

where H is the entropy function.

Therefore, using Equation 2.4 we get that our KL-based estimate is

$$p_x^{KL,\mu}(\vec{t}) = \exp\left(H(\vec{t})\right)\cdot p_x^{[\mu]}(\vec{t})^{\frac{1}{|\vec{t}|}}. \tag{2.6}$$

In the re-ranking setting (discussed in Chapters 5, 6, 7 and 8), high entropy of a document language model can (sometimes) provide an indication for the relevancy of the corresponding document. (We further discuss this issue in Chapter 6.)

The cross-entropy and the KL divergence (as described in Section 2.2) were directly used in numerous approaches within the language-modeling retrieval framework (e.g., (Xu and Croft, 1999; Ng, 2000; Zhai and Lafferty, 2001b; Lavrenko and Croft, 2003)) in settings where a single (extended) query was used to rank documents or collections. In such settings, this is equivalent to ranking by the smoothed MLE estimate, as was observed by Lafferty and Zhai (2001) and as can be seen in equations 2.4 and 2.6. (Note that in these settings $\vec{t}$ is fixed to be the query.) However, this ranking equivalence does not hold in most of our settings since we want to estimate association between different units (documents, clusters) and to avoid the above described problems of length bias and numerical underflow.

Finally, we observe that both estimates just described represent *asymmetric* notions of similarity, e.g., for arbitrary $\vec{t}$ and x, $p_x^{CE,\mu}(\vec{t}) \neq p_{\vec{t}}^{CE,\mu}(x)$. This asymmetry is (conceptually) highly important for the graph-based framework we present in Chapter 6.

Chapter 3
Exploiting corpus-structure analysis for ad hoc retrieval

We open this chapter with a toy example that provides some intuition for the potential in exploiting corpus structure — via clusters of similar documents — for ad hoc retrieval. We then survey past work on ad hoc IR that exploits information from document clusters created offline. (Chapter 5 focuses on exploiting query-dependent clusters.). We continue with an overview of some well known methods for modeling corpus structure, some of which were used in a retrieval context. We then observe that while the combination of language models and clustering was shown to be successful in tasks such as distributional clustering (Pereira et al., 1993; Rooth et al., 1999; Clark and Weir, 2002) (and its application to text categorization (Baker and McCallum, 1998)), topic detection and tracking (Spitters and Kraaij, 2001) and modeling long distance dependencies in text (Iyer and Ostendorf, 1999), there has been almost no work (until very recently) on utilizing this combination for ad hoc IR.

3.1 How can clustering help ad hoc retrieval ?

One way to model corpus structure is to cluster the documents in the corpus in a query-independent fashion (i.e., before any query is initiated). Then at retrieval time, information drawn from these clusters can potentially enhance retrieval effectiveness. In what follows we illustrate the potential of such an approach using a toy example.

Let $q = \{truck, bus\}$ be a query, and consider the documents in the following table:

Document Id	Document Terms
x_1	school, bus, classes, teachers
x_2	school, classes, teachers, study
x_3	school, teachers, classroom, grades
y_1	bus, taxi, boat, bike
y_2	taxi, boat, truck, scooter
y_3	boat, horse, taxi, bike

For simplicity assume that we represent both the documents and the query as vectors of terms (i.e., we use a vector space representation (Salton, 1968)), and that the weight for each term in a vector is its frequency within the corresponding document (or query).[1] For similarity measure we use the inner product. (Note that length normalization has no effect since all documents are of the same length.)

Now, if we rank the documents in response to q and in doing so assume that ties are broken arbitrarily we might end up with the following ranking:

$$R^{doc} = x_1, y_1, y_2, x_2, x_3, y_3.$$

(x_1 is the top retrieved document.)

However, since it is reasonable to assume that the underlying topic of the query is "vehicles" rather than "school", we would like to have y_1, y_2 and y_3 ranked as high as possible in the list.

Note that for $i \neq j$, documents x_i and y_j do not have more than one term in common, while any two documents x_i and x_j (and similarly y_i and y_j) have two terms in common, so, clustering the document collection using a *hard clustering scheme* — clustering in which each document belongs to exactly one cluster — into two clusters (e.g., using agglomerative single-link clustering (Hartigan, 1975)), could result in the following clusters: $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_3\}$.

Now, if we treat a cluster as the (long) document that results from the concatenation of the cluster's constituent documents we get the following ranking for clusters in response to q:

$$R^{clust} = Y, X.$$

Assume that to derive a document-based ranking from this cluster-based ranking we replace each cluster with its constituent documents (ordered by some criterion). We then get that y_1, y_2 and y_3 are ranked higher than the rest of the documents — our desired result. In particular, y_3, which does not contain any of the terms that occur in q, is now ranked higher than x_1, which does contain one query term, but does not seem to discuss the query topic.

This toy example demonstrates the potential effectiveness of cluster-based retrieval. Documents that discuss the same "topic" underlying the query are ranked high even if they do not exhibit high surface similarity to the query. Furthermore, if we consider the terms "boat, taxi, bike and bus" to be synonyms in the "topic space", then this example shows how the problem of synonymy is ameliorated using cluster-based retrieval.

Another interesting observation with regard to the example above is the potential superiority of cluster-based methods to pseudo-feedback (PF) methods in certain settings. Suppose we use some PF method on top of the ranking R^{doc} to define an expanded query model. If we use only the single highest-ranked document (x_1) then the model defined will be biased towards retrieving the documents x_1, x_2, x_3 instead of y_1, y_2 and y_3. Similarly, if we use the two highest-ranked

[1] While for simplicity reasons we use vector-space representation, we hasten to point out that all the conclusions we present based on this example also hold if language models are utilized.

documents x_1, y_1 then the defined model will represent a mixture of "topics" from clusters X and Y and would not necessarily result in the desired ranking.

In the next section we describe past approaches to retrieval that exploit information drawn from clusters created offline. Most of these approaches were based on the vector space model, as they were presented a long time before language models were first used for information retrieval.

3.2 Ad hoc IR using clusters created offline

Using clusters created in a query-independent fashion for the task of ad hoc retrieval has a long history. While the initial motivation for using clusters was to increase efficiency (Salton, 1971b; Croft, 1978; Voorhees, 1986) (i.e., reduce retrieval time by considering only subsets of the documents in the corpus as candidates for retrieval), effectiveness (as manifested in the quality of the retrieved results) was also an important consideration (Jardine and van Rijsbergen, 1971; Croft, 1980).

The premise that clusters can enhance retrieval effectiveness goes back to Rijsbergen's *cluster hypothesis* (van Rijsbergen, 1979): *Closely associated documents tend to be relevant to the same requests.* Some tests to evaluate the hypothesis with respect to different definitions of "closely associated" were designed (Jardine and van Rijsbergen, 1971; Voorhees, 1985; El-Hamdouchi and Willett, 1987) but different data sets exhibited different "degrees" (according to the specific choice of test) to which the hypothesis held.

In most work on cluster-based retrieval, hierarchical clustering methods were employed (Jardine and van Rijsbergen, 1971; Croft, 1980; Voorhees, 1985), as partition-based clustering approaches were shown to result in performance somewhat inferior to that of standard document-based retrieval (Salton, 1971b). In these projects, clusters were used to select documents for retrieval. To find clusters in the hierarchy in response to a query, both top-down and bottom-up search techniques were used; the latter were shown to be more effective than the former (Croft, 1980). In a later study, where different stopping criteria for the search of the clustering tree were explored, El-Hamdouchi and Willett (1989) found that simply ranking the "bottom level clusters" — those containing documents that first enter the cluster hierarchy — resulted in better performance than a bottom-up search with any of the studied stopping criteria.

To present a user with documents in response to a query, often, clusters were retrieved in their entirety, with no partial ordering imposed on their constituent documents. (In some cases a threshold on the query-document similarity was used to select documents within clusters.) Later work by Voorhees (1985) suggested a method wherein the relative ordering of documents within a cluster is determined by the similarity of the documents to the query. Another example of this approach, termed *CQL*, was presented by Liu and Croft (2004). However, CQL utilizes query-dependent clusters, not clusters created offline.

As for information representation, the leading paradigm is the vector space representation (Salton et al., 1975), and clusters were usually represented by the centroid of their constituent documents. Symmetric similarity functions (such as the cosine or Dice coefficient) were used to rank clusters and documents against queries. Furthermore, once a clustering was determined, the strength of association between a cluster and its constituent documents was not modeled — an important point we address in our work in Chapter 4.

The results of comparisons between cluster-based and document-based retrieval were inconclusive. While in certain cases cluster-based retrieval resulted in higher precision than document-based retrieval, this was not always the case and sometimes cluster-based retrieval actually hurt performance.

Several important observations, which shed some light on the body of work described above, were presented by Griffiths et al. (1986). They observed that the optimal cluster size (when using hierarchical clustering) for effective retrieval should be relatively small. (In many cases clusters contained a document and its single nearest neighbor). Following this observation, they proposed a nearest-neighbor approach to clustering but found that the resultant retrieval performance was not better than that obtained using hierarchical clustering. However, they did find that the documents retrieved by a cluster-based approach using nearest-neighbor clusters were different than the ones retrieved by a standard document-based ranking and left the question of how to combine the two approaches open, postulating that this would be highly beneficial. Similarly, El-Hamdouchi and Willett (1989) found that clusters of size two — a document and its nearest neighbor — were indeed more effective than agglomerative clusters, thereby strengthening the conclusion about the way corpus structure should be modeled (i.e., small clusters that represent local information in the corpus) when clusters are used for selecting potential relevant documents.

Our choice of modeling corpus structure using nearest-neighbors clusters in Chapter 4, along with an algorithmic framework for IR that combines cluster-based LMs with document-based LMs, could be (conceptually) regarded as following the line and open questions that Griffiths et al. (1986) established. However, apart from several aspects that our algorithmic framework addresses (e.g., modeling cluster-document association) and that were not addressed in previous work on cluster-based retrieval (including that of Griffiths et al. (1986)), we show that not only is it beneficial to model corpus structure using relatively small clusters, but the overlap of such clusters can itself be an important source of information for the retrieval process, whether clustering is performed in a query-independent fashion (Chapter 4) or query-dependent clusters are utilized (Chapter 5).

3.2.1 Soft clustering

Heretofore we surveyed work that exploited hard clustering schemes for retrieval. As we mentioned in the previous section, our framework (Chapter 4) is based on overlapping nearest-neighbor clusters for modeling corpus structure. Naturally, one of the benefits of *probabilistic* (soft) clustering methods (by definition) is that they allow documents to be associated with multiple clusters, thereby potentially modeling different similarities. Case in point: a document could discuss multiple topics (aspects) and if one assumes that clusters represent topics, then the document should be associated with the corresponding clusters.

Indeed, probabilistic clustering methods were shown to be highly effective in tasks such as distributional clustering of English words (Pereira et al., 1993; Lee and Pereira, 1999; Rooth et al., 1999; Clark and Weir, 2002) and modeling long distance dependencies in text (Iyer and Ostendorf, 1999).

One of the most well-known methods for modeling corpus structure is Latent Semantic Indexing (LSI) (Deerwester et al., 1990), which uses a vector space representation (Salton et al., 1975). LSI is based on a Singular Value Decomposition (SVD) of the term-document matrix of the corpus and results in a lower-dimensionality representation of both terms and documents. (Note that there are also probabilistic interpretations of LSI (Papadimitriou et al., 2000; Azar et al., 2001).) For the task of ad hoc retrieval, queries are "folded" into the lower dimensional space and similarity to the documents is then estimated in this space. When it comes to evaluating the empirical performance of LSI in the ad hoc retrieval setting, results are inconclusive. While in several instances LSI outperforms a basic retrieval approach (Deerwester et al., 1990), in other cases this is not the case and it is outperformed by the latter (Bast and Majumdar, 2005). Furthermore, the performance of LSI is known to be relatively sensitive to the number of dimensions used. Alternative models for creating a lower dimensional representation of the corpus has been proposed that deal with some of the problems inherent in LSI (Ando and Lee, 2001; He et al., 2004; Bast and Majumdar, 2005; Cai and He, 2005).

Hofmann (2001) proposed the probabilistic Latent Semantic Analysis (pLSA) method for modeling underlying topics in a corpus using a probabilistic model. The advantage of such a model is that it results in language models that describe the presumed topics and that could be potentially incorporated into the language modeling framework. However, for ad hoc retrieval, queries in Hofmann (2001) are "folded" into the lower dimensional space and the resultant probabilities are used as weights in a vector-space-model representation rather than in a probabilistic framework for retrieval.

Another well-known method for probabilistic analysis of topics in a corpus is *Latent Dirichlet Allocation (LDA)* (Blei et al., 2003). This model provides an improved probabilistic-based mechanism for modeling topics in a corpus. However, Blei et al. (2003) do not propose a retrieval framework that integrates the LDA-topic language models.

Note that the latter two approaches (pLSA and LDA) could be considered as clustering methods that produce probabilistic (or soft) clusters and that describe these clusters using their term distributions (i.e., language models).

3.3 Combining clusters and language models in ad hoc retrieval

In several tasks, the combination of clusters and language models has proven to be highly effective. Examples include distributional clustering (Pereira et al., 1993; Rooth et al., 1999; Clark and Weir, 2002) (and its application to text categorization (Baker and McCallum, 1998)), distributed retrieval (Xu and Croft, 1999), modeling long distance relationships in text (Iyer and Ostendorf, 1999) and topic detection and tracking (Spitters and Kraaij, 2001).

Now, while Hofmann's (2001) use of cluster-based language model probabilities to induce weights in a vector-space representation could be considered as one of the first attempts to actually combine clusters and language models for ad hoc retrieval, the retrieval approach is still implemented using the vector-space model for representation and similarity induction between the query and the documents.

Simultaneously, Azzopardi et al. (2004), Kurland and Lee (2004), and Liu and Croft (Liu and Croft, 2004) showed that integrating cluster-based information (induced from a clustering performed offline) into the language modeling framework results in improved performance with respect to the basic LM approach.

Azzopardi et al. (2004) showed that representing a document as a mixture of LDA topics and using this representation in the two stage language-model framework of Zhai and Lafferty (2002) can improve the retrieval performance with respect to that of the standard LM approach.

Liu and Croft (2004) showed that clustering the corpus offline using the k-means clustering algorithm, and smoothing a document's language model with the language model constructed from the single cluster to which the document belongs, results in better performance than that of the basic LM approach.

In Chapter 4 (based on Kurland and Lee (2004)) we present an algorithmic framework that utilizes language models created from overlapping nearest-neighbor clusters. One of the algorithms we present can be conceptually viewed as a generalization of the model suggested by Liu and Croft (2004), and we study its performance when implemented either with our nearest-neighbor clusters or k-means clusters. Furthermore, this algorithm could also be conceptually viewed as a generalization of a recent model presented by Wei and Croft (2006) that smoothes a document language model with LDA-based cluster LMs, and uses these in the query likelihood model, posting performance improvements over the basic LM approach.

Thus, the approaches proposed by Azzopardi et al. (Azzopardi et al., 2004), Kurland and Lee (2004), and Wei and Croft (2006) model corpus structure using soft (overlapping) clusters, while Liu and Croft (2004) use hard clustering for modeling the corpus structure.

Very recently, Tao et al. (2006) showed that smoothing a document language model using counts of terms in the document's nearest neighbors results in retrieval performance superior to that obtained by using a standard collection-based smoothing technique. Similarly, one of our retrieval algorithms in Chapter 4 is based on smoothing a document language with the language models of its nearest neighbors.

The work described in this chapter, and our algorithmic framework in the next chapter, utilize clusters created offline. In Chapter 5 we survey work utilizing clusters created from retrieved results (a.k.a. query-dependent clusters), and present new algorithms for exploiting such clusters in Chapters 5 and 7.

Chapter 4
Ad hoc retrieval using language models of clusters created offline[1]

In this chapter we present a novel algorithmic framework for retrieval that exploits cluster-based language models.

As described in chapter 3, clusters are a convenient representation of similarity whose potential for improving retrieval performance has long been recognized (Jardine and van Rijsbergen, 1971; van Rijsbergen, 1979; Croft, 1980). One key advantage is that they can provide smoothed, more representative statistics for their elements, as has been recognized in statistical natural language processing for some time (Brown et al., 1992). As the example in Section 3.1 shows, we could infer, for example, that a document not containing some query terms is still relevant if the document belongs to a cluster whose other constituent documents do contain these terms.

Our proposed framework is based on clusters created offline. One potential drawback of using such clusters is that, by definition, they are query-independent and therefore at retrieval time one has to estimate the relevance of the information they provide to the information request specified in the query. Also, cluster statistics may over-generalize with respect to specific member documents. In Chapters 5 and 7 we present retrieval algorithms that utilize query-dependent clusters, some of which are adaptations of instantiations of the framework presented in this chapter.

Our framework therefore incorporates both corpus-structure information — using pre-computed, overlapping clusters — and individual-document information. Importantly, although (in this chapter) cluster formation is query-independent, within our framework the *choice* of which clusters to incorporate *can* depend on the query. We consider several of the many possible algorithms arising as specific instantiations of our framework. These include both novel methods and, as special cases, both the standard, non-cluster-based LM approach (i.e., the query likelihood model) and a variant of the cluster-based *aspect model* (Hofmann and Puzicha, 1998).

Our empirical evaluation consists of experiments in an array of settings created by varying several parameters and meta-parameters; these include the corpus, the clustering scheme (nearest-neighbors vs. k-means) and the feature selection method (stopword removal and stemming). In addition, we show that a subset of our algorithms can be effective even if we employ a vector-space information representation instead of language models. Analyzing the performance results,

[1] This chapter is based on a paper written with Lillian Lee (Kurland and Lee, 2004) that appeared in the proceedings of SIGIR 2004. Portions of this chapter also represent work that appeared in a paper written with Lillian Lee and Carmel Domshlak (Kurland et al., 2005), which appeared in the proceedings of SIGIR 2005.

we find that even the worst-performing of our novel algorithms is superior to the basic LM approach, and our best performing method (the interpolation-t algorithm) consistently improves on the LM approach in a statistically significant way. Furthermore, the latter algorithm, with minimal tuning, is shown to have comparable performance with that of highly optimized state-of-the-art pseudo-feedback methods.

4.1 Notational conventions

We use d, q, c and $\mathcal{C}$ to denote a document, query, cluster, and corpus, respectively. A fixed vocabulary is assumed. We use the notation $p_d(\cdot)$, $p_c(\cdot)$ for the language models induced from d and c respectively. (Section 4.4.1 provides specific estimation details.)

It is convenient to use *Kronecker delta* notation $\delta[s]$ to set up some definitions. The argument s is a statement; $\delta[s] = 1$ if s holds, 0 otherwise.

4.2 Retrieval framework

As noted above, when we rank documents with respect to a query, we desire per-document scores that rely both on information drawn from the particular document's contents and on how the document is situated within the similarity structure of the ambient corpus.

Structure representation via overlapping clusters As discussed in Chapter 3, document clusters are an attractive choice for representing corpus similarity structure (van Rijsbergen, 1979, chapter 3). Clusters can be thought of as *facets* of the corpus that users might be interested in. While a deep analysis of facets in the corpus would probably call for clustering over different axes (e.g., document style, author), traditional work on cluster-based retrieval has mainly focused on hard clustering schemes implemented over bag-of-words representations of documents (see Jardine and Rijsbergen (1971), Croft (1980), and Vorhees (1985) for examples). In section 4.4.5 we report the empirical performance of one of our algorithms when implemented with one such scheme, the k-means clustering method.

However, given that a particular document can discuss several topics, and be relevant to a user for several reasons, or to different users for different reasons, we believe that a set of overlapping (or even probabilistic, e.g., Hofmann (2001), and Blei et al. (2003)) clusters forms a better model for similarity structure than a partitioning of the corpus. Furthermore, employing intersecting clusters may reduce information loss due to the generalization that hard clustering can introduce (van Rijsbergen, 1979, pg. 44). Such generalization might be incurred by the association of each document with a single cluster, an association that binds the document to a single topic or facet and thereby ignores other facets and topics that might also relate to the document. In light of the potential benefits of using overlapping clusters for modeling corpus structure, we will assume the use of such clusters (unless otherwise specified) throughout the rest of this chapter.

Information representation We use language models induced from documents and clusters as our information representation. Each document d (or cluster c) is represented as a distribution over single terms in the vocabulary, i.e., a term language model. We then use $p_d(q)$ and $p_c(q)$ to specify our initial knowledge of the relation between the query q and a particular document d or cluster c, respectively. (However, Section 4.4.6 shows that using a tf.idf representation also yields performance improvements with respect to the appropriate baseline, though not to the same degree as using language models does.)

Information integration To assign a ranking to the documents in a corpus $\mathcal{C}$ with respect to q, we want to score each $d \in \mathcal{C}$ against q in a way that incorporates information from the query-relevant corpus facets to which d belongs. While we create $Cl(\mathcal{C})$, the set of clusters, in advance, and hence in a query-independent fashion, to compensate, at retrieval time we base the *choice* of appropriate pre-computed clusters on the query.

How might cluster information be used? Our discussion above indicates that clusters can serve two roles. Insofar as they approximate facets of the corpus, they can aid in the *selection* of relevant documents: we would want to retrieve those that belong to clusters corresponding to facets of interest to the user. On the other hand, clusters also have the capacity to *smooth* individual-document language models, since they pool statistics from multiple documents. Finally, we must remember that over-reliance on $p_c(q)$ can over-generalize by failing to account for document-specific information encoded in $p_d(q)$.

These observations motivate the algorithm template shown in Figure 4.1. This template is fairly general: both the standard language-modeling approach introduced by Ponte and Croft (1998) and the *aspect* model (Hofmann and Puzicha, 1998) are concrete instantiations. In the template, the choice of $\text{Facets}_q(d)$ (line 4) corresponds to utilizing clusters in their selection role. The scoring step (line 5) can be thought of as integrating $p_d(q)$ with cluster-based language models in their smoothing role. The optional re-ranking step (line 7) is used as a way to further bias the final ranking towards document-specific information, if desired. Note that re-ranking can change the average non-interpolated precision but not the absolute precision or recall of the retrieval results; we therefore use it, when necessary, to enhance average (non-interpolated) precision. (Section 4.4 reports experiments studying its efficacy.)

In the next section, we describe a number of specific algorithms arising from this template, concentrating on their degree of dependence on cluster-induced language models.

```
1. *Offline*: Create $Cl(\mathcal{C})$
2. *Given $q$ and $N$, the number of documents to retrieve*:
3.     For each $d \in \mathcal{C}$,
4.         Choose a cluster subset $\text{Facets}_q(d) \subseteq Cl(\mathcal{C})$
5.         Score $d$ by a weighted combination of $p_d(q)$ and
                 the $p_c(q)$'s for all $c \in \text{Facets}_q(d)$
6.     Set TopDocs($N$) to the rank-ordered list of $N$ top-
          scoring documents
7.     Optional: re-rank $d \in \text{TopDocs}(N)$ by $p_d(q)$
8.     Return TopDocs($N$)
```

Figure 4.1: Algorithm template.

Table 4.1: Algorithm specifications. ("Top" stands for $\text{TopClusters}_q(m)$.)

	$\text{Facets}_q(d)$	Score	Re-rank by $p_d(q)$?
LM	N/A	$p_d(q)$	(redundant)
basis-select	$\{\text{Cohort}(d)\} \cap Top$	$p_d(q) \cdot \delta[\lvert\text{Facets}_q(d)\rvert > 0]$	(redundant)
set-select	$\{c : d \in c\} \cap Top$	$p_d(q) \cdot \delta[\lvert\text{Facets}_q(d)\rvert > 0]$	(redundant)
bag-select	$\{c : d \in c\} \cap Top$	$p_d(q) \cdot \lvert\text{Facets}_q(d)\rvert$	no
uniform-aspect-t	$\{c : d \in c\} \cap Top$	$\sum_{c \in \text{Facets}_q(d)} p_c(q)$	yes
aspect-t	$\{c : d \in c\} \cap Top$	$\sum_{c \in \text{Facets}_q(d)} p_c(q) \cdot p_d(c)$	yes
interpolation-t	$\{c : d \in c\} \cap Top$	$\lambda \cdot p_d(q) + (1-\lambda) \sum_{c \in \text{Facets}_q(d)} p_c(q) \cdot p_d(c)$	no

4.3 Retrieval algorithms

Table 4.1 summarizes the algorithms we consider, which represent a few choices out of the many possible ways to instantiate the template of Figure 4.1. Our preference in picking these algorithms has been towards simpler methods, so as to focus on the impact of using cluster information (as opposed to the impact of tuning many weighting parameters, say).

First step: Cluster formation and selection There are many algorithms that can be used to create $Cl(\mathcal{C})$, the set of overlapping document clusters required by Figure 4.1's template (line 1). In our experiments, we simply have each document d form the *basis* of a cluster Cohort(d) consisting of d and its nearest neighbors:

Definition 1. Cohort(d) *is the cluster that contains* d *and its* $k-1$ *nearest neighbors, which are determined by measuring the Kullback-Leibler (KL) divergence between d's unsmoothed language model (as defined over single terms) and the smoothed language models (as defined over terms) of other documents in the corpus; k is a free parameter.*

Note that two clusters with different basis documents may contain the same set of documents.[2]

The first retrieval-time action specified by our algorithm template is to choose $\text{Facets}_q(d)$, a query-dependent subset of $Cl(\mathcal{C})$ (see line 4 in Figure 4.1). In all the algorithms described below except the baseline (which does not use cluster information), there is a document-selection aspect to this subset, in that only documents in some $c \in \text{Facets}_q(d)$ can appear in the final ranked-list output. Ideally, we would use the clusters best approximating those facets of the corpus that are most representative of the user's interests, as expressed by q; therefore, we require that $\text{Facets}_q(d)$ be a subset of those clusters that exhibit the strongest similarity to the query, $\text{TopClusters}_q(m)$:

Definition 2. $\text{TopClusters}_q(m)$ *is the top* m *clusters* $c \in Cl(\mathcal{C})$ *with respect to* $p_c(q)$*;* m *is a free parameter.*

But we also want to evaluate d only with respect to the facets it actually exhibits. Thus, in what follows (except for the baseline), $\text{Facets}_q(d)$ is always defined to be a subset of $\{c : d \in c\} \cap \text{TopClusters}_q(m)$; we assume m is large enough to produce the desired number of retrieved documents N.

Retrieval methods and reference comparisons

Baseline method The baseline for our experiments, denoted **LM**, is to simply rank documents by $p_d(q)$ — no cluster information is used. (See Section 4.4.1 for language-model estimation details.)

Selection methods In this class of algorithms, the cluster-induced language models play a very small role once the set $\text{Facets}_q(d)$ is selected. In essence, the standard language-modeling approach (that is, ranking by $p_d(q)$) is invoked to rank the documents comprising the clusters in $\text{Facets}_q(d)$. This method of scoring is intended to serve as a precision-enhancing mechanism, downgrading documents that happen to be members of some $c \in \text{Facets}_q(d)$ by dint of similarity to d in respects not pertinent to q.

In the **basis-select** algorithm, the net effect of the definition given in Table 4.1 is that only the *basis* documents of the clusters in $\text{TopClusters}_q(m)$ are allowed to appear in the final output list. Thus, this algorithm uses the pooling of statistics from documents in $\text{Cohort}(d)$ simply to decide whether d is worth ranking; the rank itself is based solely on $p_d(q)$.

The **set-select** algorithm differs in that *all* the documents in the clusters in $\text{TopClusters}_q(m)$ may appear in the final output list — the "set" referred to in

[2] Smoothing document language models with language models of their nearest neighbors was also presented in Ogilvie (2000) and Tao et al. (2006). A similar smoothing approach, based on local information, was employed for the query model in Lavrenko (2000) rather than to the documents' models.

the name is the union of the clusters in $\text{TopClusters}_q(m)$. The idea is that any document in a "best" cluster, basis or not, is potentially relevant and should be ranked. Again, the ranking of the selected documents is by $p_d(q)$.

Another natural variant of the same idea is that documents appearing in more than one cluster in $\text{TopClusters}_q(m)$ should get extra consideration, given that they appear in several (approximations of) facets thought to be of interest to the user. This idea gives rise to the **bag-select** algorithm, so named in reference to the incorporation of a document's multiplicity in the *bag* formed from the "multi-set union" of all the clusters in $\text{Facets}_q(d)$. Each selected document d is assigned a score consisting of the product of its language-modeling score $p_d(q)$ and the number of top clusters it belongs to.

It is important to observe that for the basis-select and set-select algorithms the re-rank step is redundant since d's score is exactly $p_d(q)$ (provided that $\text{Facets}_q(d)$ is not empty). For the bag-select algorithm, however, we do not use the additional re-rank step since the score of a document is already based on the probability $p_d(q)$ and empirical results indeed show that re-ranking does not improve performance.

Aspect methods We now turn to algorithms making more explicit use of clusters as smoothing mechanisms. In particular, we study what we term "aspect-t" methods. Our choice of name is a reference to the work of Hofmann and Puzicha (1998), which conceives of clusters as explanatory latent variables underlying the observed data. (The "t" stands for "truncated", details below). In our setting, this idea translates to using $p_c(q)$ as a proxy for $p_d(q)$, where the degree of dependence on a particular $p_c(q)$ is based on the strength of association between d and c. The **aspect-t** algorithm measures this association by $p_c(d)$; the **uniform-aspect-t** algorithm assumes that every $d \in c$ has the same degree of association to c (further details below).

The scoring function we propose can be motivated by appealing to the probabilistic derivation of the aspect model (Hofmann and Puzicha, 1998), as follows. It is a fact that, regardless of the interpretation of the following conditional probabilities, one can write

$$p(q|d) \quad = \quad \sum_c p(q|d,c)p(c|d). \tag{4.1}$$

The aspect model assumes that a query is conditionally independent of a document given a cluster (which is a way of using clusters to smooth individual-document statistics), in which case

$$p(q|d) = \sum_c p(q|c)p(c|d). \tag{4.2}$$

Our **aspect-t** algorithm then arises by replacing the conditional probabilities with the corresponding language model probabilities (e.g., changing $p(c|d)$ to $p_d(c)$)

and summing over just the clusters in $\mathrm{Facets}_q(d)$, as opposed to requiring that the summation be over all clusters[3]:

$$\sum_{c \in \mathrm{Facets}_q(d)} p_c(q) \cdot p_d(c).$$

As a point of comparison, we also introduce the **uniform-aspect-t** algorithm, which essentially corresponds to setting $p_d(c)$ to the uniform distribution. In both cases, re-ranking by $p_d(q)$ is applied, since this important quantity does not otherwise appear in the scoring function.

A hybrid algorithm The selection-only algorithms emphasize $p_d(q)$ in scoring a document d; in contrast, the scoring functions of the aspect-t algorithms make reference to the query only in the term $p_c(q)$. We created the **interpolation-t** algorithm to combine the advantages of these two approaches.

The algorithm can be derived by dropping the original aspect model's conditional independence assumption — namely, that $p(q|d,c) = p(q|c)$ — and instead setting $p(q|d,c)$ in Equation 4.1 to $\lambda p(q|d) + (1-\lambda)p(q|c)$, where λ indicates the degree of emphasis on individual-document information.

If we do so, we get that

$$p(q|d) = \sum_c [\lambda p(q|c) + (1-\lambda)p(q|d)]p(c|d)$$

and since $\sum_c p(c|d) = 1$, we have

$$p(q|d) = \lambda p(q|d) + (1-\lambda)\sum_c p(q|c)p(c|d).$$

Finally, applying the same estimation techniques as described in our discussion of the aspect-t algorithm yields a score function that is the linear interpolation of the score of the standard LM approach and the score of the aspect-t algorithm. Note that no re-ranking step occurs; as we shall see, the interpolation-t algorithm's incorporation of document-specific information yields higher precision.

Our interpolation-t algorithm can be conceptually considered as a generalization of models proposed by Liu and Croft (2004), and Wei and Croft (2006). Liu and Croft (2004) presented the *CBDM* model in which a document language model is smoothed via interpolation with the language model constructed from the single cluster to which the document belongs; the cluster is a result of partitioning the corpus using the k-means algorithm. The smoothed model is then used instead of the original document language model for ranking. This method is a special case of our interpolation-t algorithm if we set $\mathrm{TopClusters}_q(m) = Cl(\mathcal{C})$ and note that our framework does not require clusters to be overlapping.

[3] The suffix "-t" in the aspect-t methods' names stands for the truncated sum used in the scoring function.

Similarly to the work of Liu and Croft (2004), Wei and Croft (2006) proposed interpolating a document language model with language models constructed from LDA clusters (Blei et al., 2003). Again, setting $\text{TopClusters}_q(m) = Cl(\mathcal{C})$ in our framework, and in addition setting $\text{Facets}_q(d) = \text{TopClusters}_q(m)$ (as LDA clusters are represented as probabilities over terms in the vocabulary, and therefore the notion of membership of documents in clusters does not exist, in contrast to our nearest-neighbor clustering approach), we get that Wei and Croft's model is a special case of our interpolation-t algorithm with LDA clusters instead of nearest-neighbor clusters.[4] A similar LDA-based smoothing of document language models is presented by Azzopardi et al. (Azzopardi et al., 2004) wherein the two-stage language model framework is used (Zhai and Lafferty, 2002).

Similarly to the work of Liu and Croft (2004), Wei and Croft (2006), and our interpolation-t algorithm, although using a vector-space representation for documents instead of language models in the retrieval algorithm, Hofmann (2001) uses document vectors that integrate document-based information and cluster-based information that results from the pLSA algorithm, via interpolation.

As noted above, the interpolation-t scoring function is a linear interpolation of the score of the standard LM approach and the score of the aspect-t algorithm. Similarly, Lee et al. (2001) use score-based interpolation of query-document and query-cluster similarity scores, but do not model the strength of association between documents and clusters. Thus, their model — although operating in the vector-space as opposed to our algorithms that operate in the language-model space — resembles a variant of the interpolation-t algorithm in which the interpolation is between the score of the standard LM approach and the score of the uniform-aspect-t algorithm.

4.4 Evaluation

In this section we present a suite of experiments designed to examine the effectiveness of our algorithms and the different factors that affect their performance, such as parameter settings and feature extraction (i.e., stemming and stopword removal). In addition, we compare our nearest-neighbor clustering approach to a hard clustering scheme (k-means) that was suggested in previous work on cluster-based retrieval, as mentioned above. We also study the clustering properties of relevant documents in the tested corpora and examine the effect of implementing our algorithms with a vector-space representation in comparison to the language-modeling approach we have focused on throughout this chapter.

[4] We note that the models of Liu and Croft (2004) and Wei and Croft (2006) use interpolation at the "term level", while the interpolation-t algorithm utilizes interpolation at the "query level".

4.4.1 Language model induction

Our algorithms require the estimation of the following language-model probabilities: $p_d(q)$, $p_c(q)$, and $p_d(c)$. We note that these probabilities represent the strength of association between pairs of text spans (i.e., document-query, cluster-query and document-cluster respectively). Therefore, to avoid the length-bias and underflow problems described in Section 2.3, we use the estimate from Equation 2.3 (page 15) for the values of $p_d(q)$ and $p_c(q)$. Following the derivation given in Section 2.3, we get the estimates $p_d^{[\mu]}(q)^{\frac{1}{|q|}}$ and $p_c^{[\mu]}(q)^{\frac{1}{|q|}}$ respectively, which are the length-normalized (Dirichlet-smoothed-)language-model probabilities.

In order to have the estimator for document-cluster association probability ($p_d(c)$) form a valid probability distribution, i.e., to have $\sum_{c:c\ni d} p(c|d) = 1$ for every document d, we use the following normalized estimate for $c \ni d$:[5]

$$\frac{p_d^{[\mu]}(c)^{\frac{1}{|c|}}}{\sum_{c_i \ni d} p_d^{[\mu]}(c_i)^{\frac{1}{|c_i|}}}.$$

Note that such normalization implies that a document can only be associated with the clusters to which it belongs, and the document-cluster strength of association is the relative language-model similarity between the document and the cluster with respect to all the clusters to which the document belongs.

The Dirichlet smoothing parameter μ was set to the value 2000 recommended by Zhai and Lafferty (2001b).

4.4.2 Experimental setup

Our main experiments were conducted on TREC data. We used titles (rather than full descriptions) as queries, resulting in an average length of 2-5 terms. Some characteristics of our three corpora are summarized in the following table.

corpus	# of docs	queries	# relevant documents
AP89	84,678	1-46,48-50	3261
AP88+89	164,597	101-150	4850
LA+FR	187,526	401-450	1391

Following Zhai and Lafferty (2001b), query 47 was omitted from AP89. Note that LA+FR is more heterogeneous than the other two corpora, as it contains both newswire and US government rules, notices, and related documents.

We used the Lemur toolkit (www.lemurproject.org) to run our experiments. The number of documents to be returned at the end of the retrieval process was fixed at $N = 1000$. In the results reported below, the interpolation parameter λ for the interpolation-t algorithm was determined by optimizing for average non-interpolated precision at N. (The value of λ was chosen from $\{0.1,\ldots,0.9\}$.) For

[5] Experiments showed that using the raw estimate (without normalization) results in inferior performance.

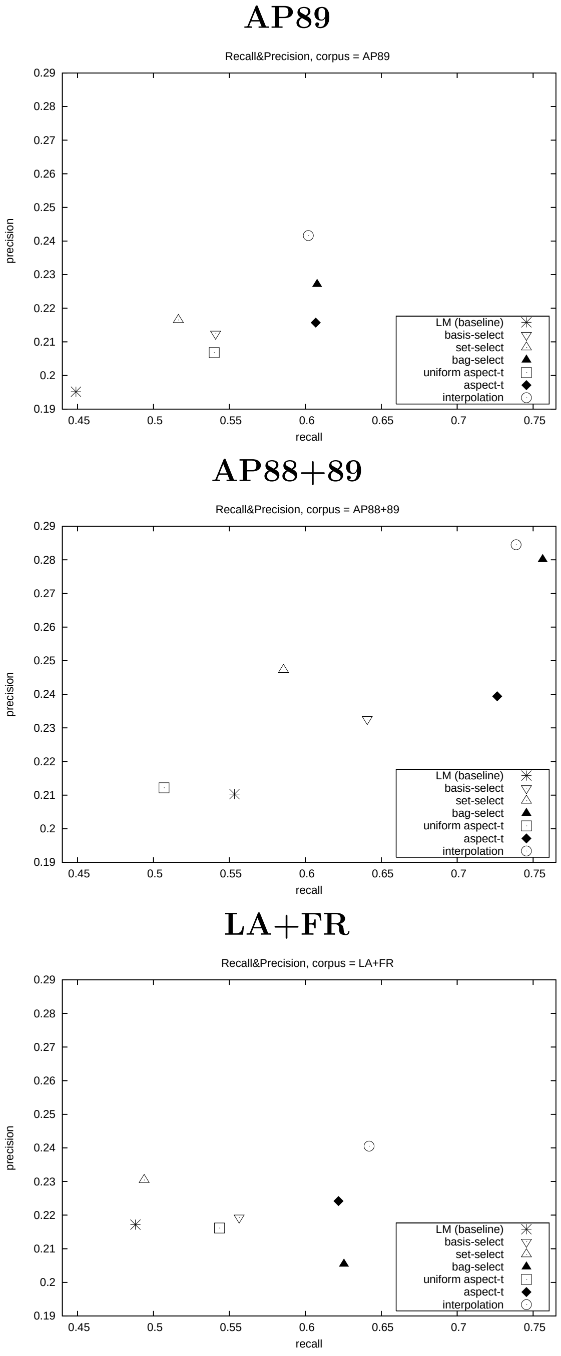
the selection algorithms, we also optimized the value of m, the number of top clusters retrieved, with respect to average non-interpolated precision; for the aspect and interpolation-t algorithms, we experimented with two different methods for choosing m, reported below. After limited experimentation with its effect on average precision, the cluster size k was fixed at 40 for AP89, 20 for AP88+89, and 10 for LA+FR.

Figure 4.2: Comparison of algorithms under the following conditions: (1) the number of clusters for the selection algorithms was chosen to optimize average precision at $N = 1000$, but for the aspect and interpolation-t algorithms, all clusters were used; and (2) no stemming or stopword removal was applied.

Table of performance numbers. Italics indicate results superior to those of the baseline language model; b's and i's mark results that differ to a statistically significant degree from the baseline or the interpolation algorithm, respectively; and bold indicates the best results for a given experimental setting (column).

	AP89		AP88+89		LA+FR	
	prec	recall	prec	recall	prec	recall
LM (baseline)	19.52%	44.89%	21.03%	55.34%	21.72%	48.81%
basis-select	*21.31%*^{bi}	*54.06%*^{bi}	*23.27%*^{bi}	*64.22%*^{bi}	*21.84%*^{i}	*54.92%*^{i}
set-select	*20.12%*^{bi}	*49.68%*^{i}	*23.13%*^{bi}	*69.57%*^{bi}	*22.33%*^{bi}	*57.30%*^{bi}
bag-select	*22.71%*^{b}	**60.78%**^{b}	*28.01%*^{b}	**75.63%**^{b}	20.54%^{i}	*62.54%*^{b}
uniform-aspect-t	*20.68%*^{bi}	*54.00%*^{i}	*21.22%*^{i}	50.70%^{bi}	21.62%^{i}	*54.35%*^{i}
aspect-t	*21.57%*^{bi}	*60.69%*^{b}	*23.94%*^{bi}	*72.63%*^{bi}	*22.42%*^{bi}	*62.19%*^{b}
interpolation-t	**24.16%**^{b}	*60.20%*^{b}	**28.45%**^{b}	*73.88%*^{b}	**24.05%**^{b}	**64.20%**^{b}

Figure 4.2 (continued)


AP89

AP88+89

LA+FR

Graphical version of the previous page table's results. Triangles indicate selection algorithms; quadrilaterals represent aspect algorithms, and the circle indicates the hybrid interpolation-t algorithm. Both axes of all three plots are to the same scale.

In general, we report both average non-interpolated precision and recall, although any optimizations were only performed with respect to the former evaluation metric. To compute statistical significance, we used the Wilcoxon two-sided test with significance threshold $p = 0.05$ and with the degrees of freedom dependent on the number of queries for the given corpus. (Refer back to Section 1.2 for more details regarding our evaluation methodology.)

4.4.3 Primary evaluations

Our first set of comparisons were performed under the following experimental conditions. No stemming or stopword removal was performed at this stage. For the aspect and interpolation-t algorithms, we set m, the number of top clusters, to its maximum value (namely, the number of documents in the corpus), in deference to the original probabilistic motivation for the aspect model given in Equation 4.2 above. (Note that by our definitions of $\text{Facets}_q(d)$ for these methods, setting m to the size of the corpus means that the set of clusters considered in scoring a particular document d is precisely the set of clusters containing d.) For the selection algorithms, on the other hand, we searched for the value of m optimizing average precision at N, as mentioned above, since limiting the number of clusters used is, conceptually, an important part of the process of selecting documents.

Figure 4.2 shows that all of our algorithms almost always provide better average precision than the baseline language model, with many of the performance enhancements being statistically significant. Furthermore, recall at the settings that optimize precision is also much better than the basic language model's. Additionally, Table 4.2 shows that similar patterns of behavior are observed if stemming (via the porter stemmer) and removal of INQUERY stopwords (Allan et al., 2000) are applied (this pre-processing does generally improve the performance of all algorithms, including the baseline). Thus, it appears that there is a great deal of useful information carried by our language-model-based nearest-neighbor clusters that can be incorporated in a variety of ways to useful effect.

We now consider differences between individual algorithms; for the sake of brevity, we examine only the results shown in Figure 4.2. The interpolation-t algorithm always exhibits the best precision, with the corresponding recall rates also being very high, relatively speaking. The bag-select algorithm can also yield very good results, although not consistently so: on AP89 and AP88+89, it posts very high precision (indeed, statistically indistinguishable from that of the interpolation-t algorithm on those corpora), but it is the lowest-precision algorithm on LA+FR. The set-select yields precision that is somewhat better than that of the uniform-aspect-t algorithm, but in several cases inferior to that of the basis-select and aspect-t algorithms. The precision of the basis-select algorithm tends to lie between that of the two aspect algorithms. Of the aspect algorithms, aspect-t yields better results than uniform-aspect-t, which is presumably an effect of the fact that the aspect-t algorithm is able to down-weight those clusters that a document being scored belongs to only in a weak sense.

Table 4.2: Version of the comparison results in Figure 4.2 in which stemming and removal of stopwords was applied to the corpora first. Thus, the number of clusters for the selection algorithms was chosen to optimize average precision at N, but for the aspect and interpolation-t algorithms, all clusters were used. Italics indicate results superior to those of the baseline language model; b's and i's mark results that differ to a statistically significant degree from the baseline or the interpolation algorithm, respectively; and bold indicates the best results for a given experimental setting (column).

	AP89		AP88+89		LA+FR	
	prec	recall	prec	recall	prec	recall
LM (baseline)	20.77%	47.99%	23.55%	66.10%	24.38%	64.20%
basis-select	*22.09%*^{bi}	*55.11%*^{bi}	*25.44%*^{bi}	*71.18%*^{i}	*24.42%*^{bi}	*65.13%*^{bi}
set-select	*21.31%*^{bi}	*52.87%*^{bi}	*25.64%*^{bi}	*76.73%*^{bi}	*24.59%*^{i}	*64.99%*^{i}
bag-select	*24.36%*^{b}	*61.70%*^{b}	*29.11%*^{b}	*80.65%*^{b}	21.43%^{bi}	*67.07%*
uniform-aspect-t	*21.89%*^{b}	*56.42%*	*24.11%*^{i}	59.44%^{bi}	23.20%^{i}	53.56%^{bi}
aspect-t	*22.61%*^{bi}	*61.85%*^{b}	*26.48%*^{bi}	*79.04%*^{bi}	*24.66%*^{bi}	*66.00%*^{i}
interpolation-t	**25.41%**^{b}	**62.16%**^{b}	**29.96%**^{b}	**80.81%**^{b}	**27.06%**^{b}	**75.34%**^{b}

Table 4.3: Comparison between bag-select and simply counting the overlap between top clusters, optimizing the parameter m (the number of top clusters retrieved) for average precision. A "b" marks a statistically significant difference from the baseline. Bold highlights the best performance for a given experimental setting (column).

	AP89		AP88+89		LA+FR	
	prec	recall	prec	recall	prec	recall
LM (baseline)	19.52%	44.89%	21.03%	55.34%	**21.72%**	48.81%
bag-select	**22.71%**^{b}	**60.78%**^{b}	**28.01%**^{b}	**75.63%**^{b}	20.54%	**62.54%**^{b}
overlap only	15.54%	*59.12%*^{b}	*23.38%*	*73.19%*^{b}	10.85%^{b}	*58.95%*^{b}

It is interesting to note that bag-select (score function: $p_d(q) \cdot |\text{Facets}_q(d)|$) performs so differently from set-select (score function: $p_d(q) \cdot \delta[|\text{Facets}_q(d)| > 0]$). The two algorithms can produce different results only because we are using (highly) overlapping clusters. It seems that for our selection methods, utilizing the exact degree of overlap is helpful on AP89 and AP88+89, but not on LA+FR. Incidentally, the performance that results from just counting the overlap among the top-retrieved clusters — that is, employing the score function $|\text{Facets}_q(d)|$ (without subsequent re-ranking by $p_d(q)$) — is shown in Table 4.3. As can be seen, ignoring the query-generation probability $p_d(q)$ is clearly detrimental to performance.[6]

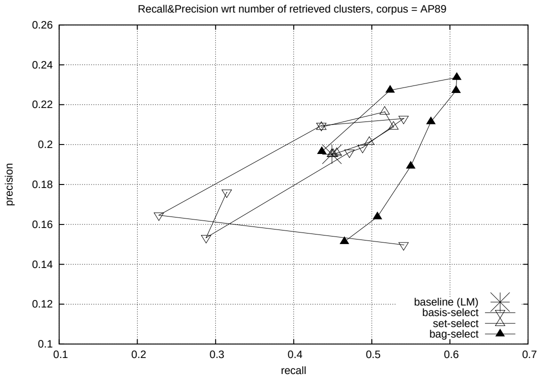
[6] Degradation in precision also occurs if we take the basis-select algorithm and remove the $p_d(q)$ term. Note that such an alteration corresponds to applying the basic language-modeling approach to the "document" Cohort(d) rather than to d itself, and so can be thought of as a smoothing method wherein the document language model is created by backing off completely to a cluster language model.

We note that both aspect-t and uniform-aspect-t resemble the bag-select method in that they tend to assign higher scores to documents that belong to more than one top cluster (thus incorporating information regarding degree of cluster overlap) and also utilize $p_d(q)$ (albeit for re-ranking rather than directly in a score function). Given this similarity, and the fact the bag-select method substantially outperforms (in terms of precision) both aspect-t and uniform-aspect-t, we need to examine whether the performance difference is due solely to the fact that the bag-select algorithm (and the selection algorithms in general) were allowed an extra free parameter in the experiments reported in Figure 4.2, namely, m, the (maximum) number of top clusters considered for each document. Therefore, we now examine the effect of varying this parameter for all our cluster-based algorithms, not just the selection ones.

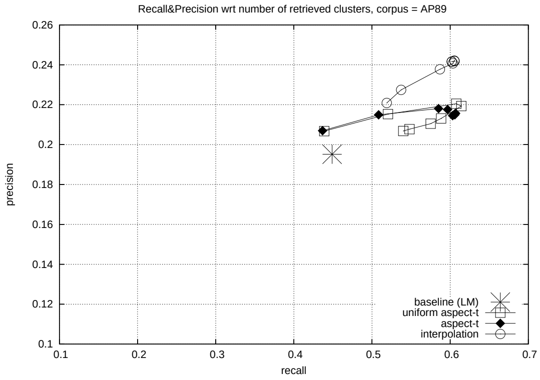
Figure 4.3 demonstrates that altering m has some interesting effects. The selection algorithms all exhibit a "clockwise" pattern: at first, increasing the maximum number of top clusters considered improves both precision and recall (after all, these methods require some minimum number of clusters — e.g., 1000 in the basis-select case — in order to be able to return $N = 1000$ documents); but too many clusters causes both precision and recall to drop. The uniform-aspect-t also follows this "clockwise" pattern, which makes intuitive sense since considering a large number of clusters per document means that clusters to which the document is only weakly associated are allowed to participate in the scoring decision. At small values of m, the aspect-t algorithm yields extremely similar results to those of the uniform-aspect-t algorithm. However, the precision of the aspect-t and interpolation-t methods does not greatly suffer when one chooses a large number of clusters (in fact, in the latter case such values are preferable), indicating that both algorithms are able to reduce the contribution of clusters that are only weakly associated with a given document in a fairly effective manner. On the other hand, the robustness of the performance of the aspect-t algorithm to choice of m means that its precision never rises appreciably above that of the best achievable by bag-select no matter what value of m is picked.

Figure 4.3: The effect of varying the number of top-retrieved clusters: $m = 50$, 100, 500, 1K, 5K, 10K, 50K, all. (x-axis: recall, y-axis: precision.) All curves move either roughly clockwise or diagonally upward as m increases. As before, triangles indicate selection algorithms, quadrilaterals represent aspect algorithms, and the circle indicates the hybrid interpolation-t algorithm. The axes for plots on the same corpus are to the same scale; thus, the baseline indicator is in the same spot in each pair of plots. No stemming or stopword removal was applied.

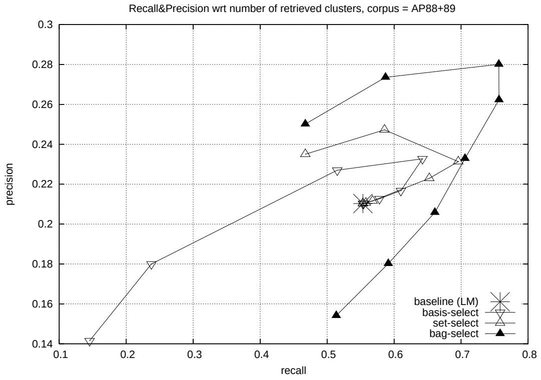
AP89, selection algorithms



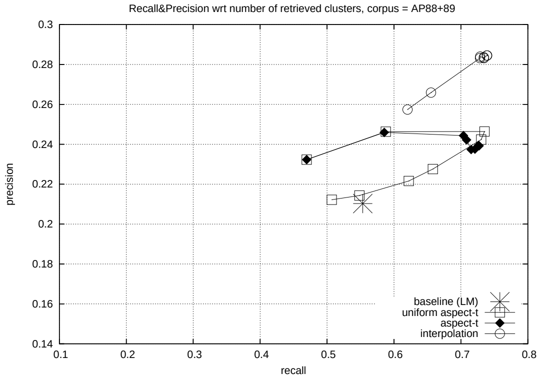
AP89, aspect & interpolation algo.



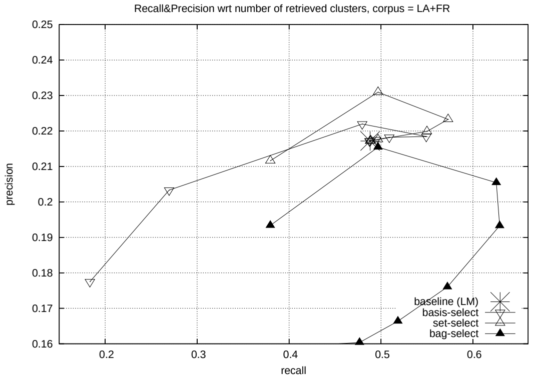
AP88+89, selection algorithms



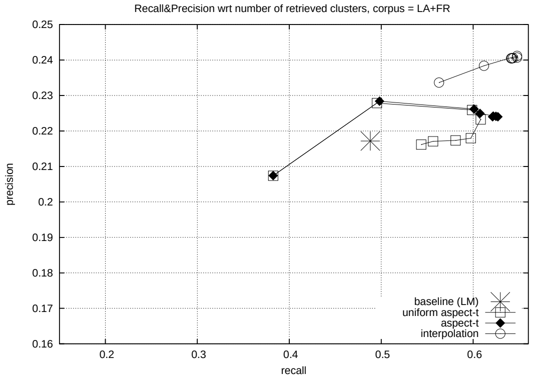
AP88+89, aspect & interpolation algo.



LA+FR, selection algorithms



LA+FR, aspect & interpolation algo.

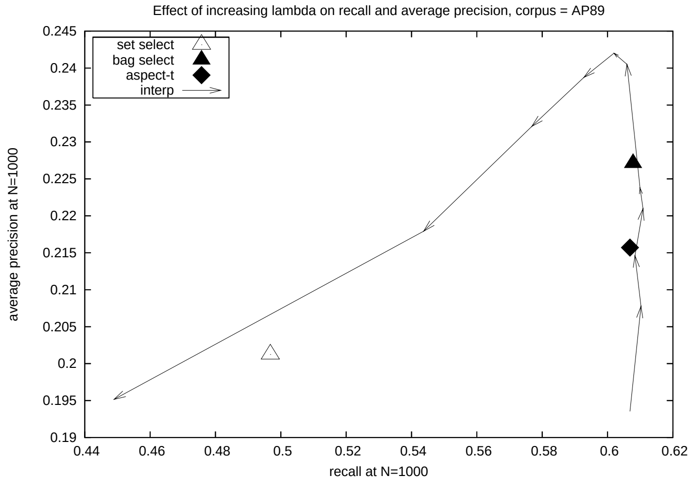


Given that the aspect-t and interpolation-t algorithms have similar scoring functions, it is interesting to observe how much the results they produce differ — the corresponding performance curves in Figure 4.3 do not even cross. One reason for the superior performance of interpolation-t might be its different handling of $p_d(q)$ information — aspect-t re-ranks the top-scoring documents by it, whereas for every document, interpolation-t linearly interpolates aspect-t's initial document score with $p_d(q)$. But again, the interpolation-t method has an extra parameter, so it could be the case that it performs better than the aspect-t algorithm simply because of overfitting. Therefore, we next study the effect of varying the interpolation parameter λ, which represents the degree of dependence on $p_d(q)$ instead of the cluster-based scoring function of the aspect-t algorithm.

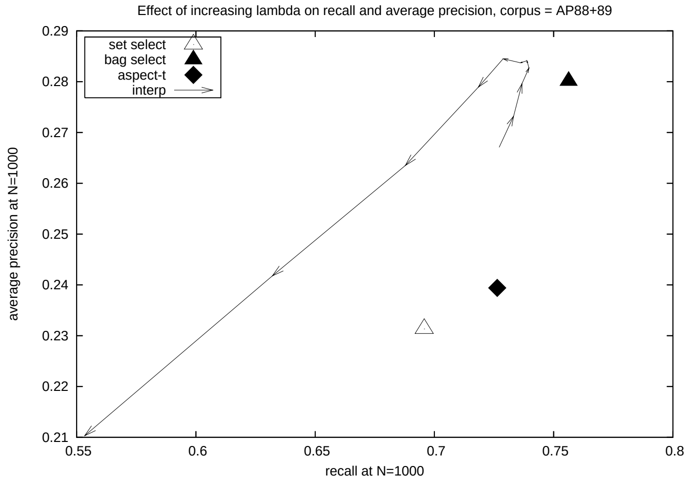
Figure 4.4 plots the "trajectory" of the interpolation-t algorithm through performance space as λ is increased. Small λ's (emphasizing clusters) produce good recall but, in the AP89 and LA+FR case, relatively poor precision. Large values of λ (emphasizing documents) always degrade recall in comparison to small values; on LA+FR, they do yield higher precision in return, but on the other two corpora, precision is either about the same as or quite a bit lower than that achieved when $\lambda = 0$. It appears that taking values from a reasonably-sized interval around 0.6 or so (and thus striking a balance between cluster and document-specific information) guarantees performance superior to that of the aspect-t algorithm, although for AP89 and AP88+89, somewhat careful tuning of λ seems to be necessary to improve over bag-select (although the latter algorithm is itself sensitive to the choice of its free parameter, m).

Figure 4.4: Performance of the interpolation-t algorithm as λ takes values in $\{0, 0.1, 0.2, \ldots, 0.9, 1\}$ while m is held fixed at its maximum value. (x-axis: recall, y-axis: precision.) Note that $\lambda = 0$ (arrow tail) corresponds to a version of the aspect-t algorithm *without* re-ranking, and that $\lambda = 1$ (arrow head) is equivalent to the baseline language model. The results of some other methods, copied over from Figure 4.2, are also depicted for reference.

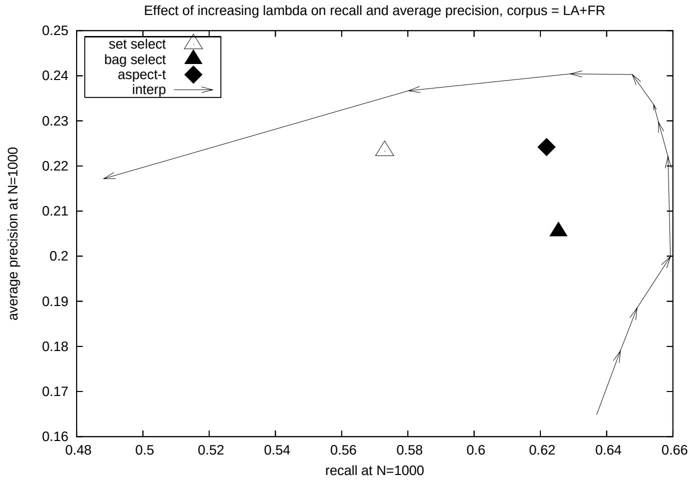
AP89



AP88+89



LA+FR



Aside: the re-rank step We note that at $\lambda = 0$, the interpolation-t algorithm corresponds to a version of the aspect-t algorithm in which re-ranking of the top N documents d by $p_d(q)$ is not applied. Figure 4.4 thus also shows that on AP89 and LA+FR, re-ranking substantially improves the average precision of aspect-t. We can therefore ask whether re-ranking might also improve the average precision of our best-performing algorithm, the interpolation-t method, even though its score function already incorporates $p_d(q)$.

Figure 4.5 shows that for intermediate to large values of λ (the value $\lambda = 0.6$, which yields near-optimal average precision on all our corpora, qualifies), re-ranking degrades performance. It is only when λ is small, so that little weight is given to document-specific language models in the interpolation-t algorithm's score function, that re-ranking can help, and in fact, in the LA+FR case, re-ranking never helps. Thus, again, interpolation seems to be superior to re-ranking as a way to balance cluster and document information.

Summary Given the set of results described throughout this section, we come to the following main conclusions. The interpolation-t method is clearly the best performing algorithm, with this being perhaps due in part to the way in which it balances document-specific information and information drawn from (multiple) clusters. But, all of the algorithms we have proposed for integrating cluster and document language models almost always provide better results than our baseline of document-based language-modeling, demonstrating that the clusters themselves convey a great deal of useful information.

4.4.4 Comparison to pseudo-feedback methods

Our cluster-based algorithms utilize corpus-structure information that is induced in a query-independent fashion, by offline clustering. As described above, such information is used to represent the "context" of the documents within the corpus. In contrast, pseudo feedback (PF) methods exploit corpus structure via query-biased sampling in order to "enhance" the original query with "corpus context". Indeed, the new query model that such methods construct from the initially retrieved list of documents often represents similarities within the list. Thus, such similarities could be considered as representing corpus-structure information that pertains to the query in hand.

To compare representatives of the two paradigms for exploiting corpus-structure, we now compare our interpolation-t algorithm against two well known and highly

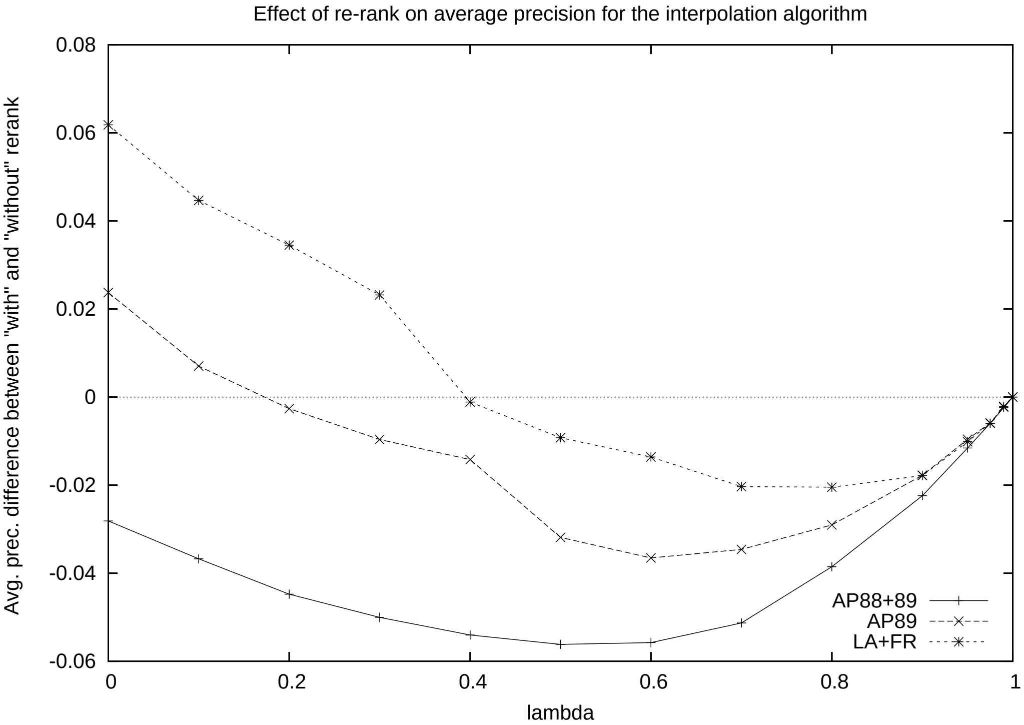


Figure 4.5: Effect on average precision of re-ranking the interpolation-t algorithm's retrieval results by $p_d(q)$, shown as a function of the interpolation parameter λ. When the function is positive, it is better to re-rank. The number of top clusters retrieved was fixed at its maximum value.

effective pseudo-feedback methods, the Rocchio algorithm (Rocchio, 1971) as applied to highest-ranked documents with respect to an initial search (using positive feedback only), and the relevance model (Lavrenko and Croft, 2001). Our implementation of both PF models is based on the Lemur toolkit (www.lemurproject.org).

Rocchio's original method is based on a vector space representation for documents and queries; we denote as $\mathbf{x}$ the vector representation of text x. We use the inner product as similarity measure in the vector space. Then, given the list of highest ranked documents (denoted as $\mathcal{D}_{\text{init}}$) with respect to an initial search performed in response to q (using document-query similarity), a new query vector is defined as follows:

$$\mathbf{q} + \alpha \frac{1}{|\mathcal{D}_{\text{init}}|} \sum_{d_i \in \mathcal{D}_{\text{init}}} \mathbf{d_i},$$

and is used to (re-)rank all the documents in the corpus.

Following past work (e.g., Lafferty and Zhai (2001)), and to comply with optimization steps that we implemented for the relevance model (details further below), we also experimented with a version of this model wherein we only use the β

terms with highest weights in the vector $\frac{1}{|\mathcal{D}_{\text{init}}|}\sum_{d_i \in \mathcal{D}_{\text{init}}} \mathbf{d_i}$ to add to $\mathbf{q}$ in the above equation.

Throughout some experiments with different forms of weighting schemes we found that using "log(tf)∗log(idf)" weights (Zobel and Moffat, 1998) results in very good performance. In our experiments, we varied the following free parameters so as to optimize performance: (i) the number of highest-ranked documents from the initial search used for feedback ($|\mathcal{D}_{\text{init}}|$), (ii) the number of terms β to augment the original query with, and (iii) the weighting coefficient α for the augmenting terms.

The relevance model (Lavrenko and Croft, 2001) takes a generative perspective: assuming that there is a single *relevance* language model $\mathcal{R}$ underlying the creation of both q and the documents relevant to q, documents are ranked by their degree of "match" with $\mathcal{R}$, rather than by how well they directly match the query. In implementation, Lavrenko and Croft estimate $\mathcal{R}$ by combining the language models of those documents assigning the highest language-model probabilities to q.

Our Lemur-based implementation of the relevance model utilized i.i.d sampling (following the implementation details described in Lavrenko and Croft (2003)) to construct $\mathcal{R}$. We set $\mathcal{D}_{\text{init}}$ to be the set of highest ranked documents with respect to a retrieval performed (over the entire corpus) using $p_d^{CE,\mu}(q)$ (setting $\mu = 2000$ as we did for our algorithms). [7] We write $q = q_1, \ldots q_{|q|}$ for the query, and estimate the unigram language model of document $d \in \mathcal{D}_{\text{init}}$ using Jelinek-Mercer (JM) smoothing (Zhai and Lafferty, 2001b), instead of the Dirichlet smoothing technique we use throughout the thesis, in order to follow the exact implementation details of Lavrenko and Croft (2001; 2003)). Thus, for term t in the vocabulary we get:

$$\widetilde{p}_d^{JM;[\lambda]}(t) \stackrel{def}{=} \lambda \widetilde{p}_d^{MLE}(t) + (1-\lambda)\widetilde{p}_{\mathcal{C}}^{MLE}(t).$$

(Refer back to Section 2.3 (page 13) for further details regarding language model notation.)

We then define $\mathcal{R}$ as

$$p_{\mathcal{R}}(t) \stackrel{def}{=} \sum_{d_i \in \mathcal{D}_{\text{init}}} \widetilde{p}_{d_i}^{JM;[\lambda]}(t) \cdot p(d_i|q), \tag{4.3}$$

where for each document $d \in \mathcal{D}_{\text{init}}$ we write:

$$p(d|q) = \frac{p(d)\prod_j \widetilde{p}_d^{JM;[\lambda]}(q_j)}{\sum_{d_i \in \mathcal{D}_{\text{init}}} p(d_i)\prod_j \widetilde{p}_{d_i}^{JM;[\lambda]}(q_j)},$$

and set $p(d_i)$ to a constant (thus assuming a uniform distribution over documents).

Given the constructed relevance language model $\mathcal{R}$, we use the divergence $D(\mathcal{R}||\widetilde{p}_d^{[\mu]}(\cdot))$ (recall that $\widetilde{p}_d^{[\mu]}(\cdot)$ is the Dirichlet-smoothed language model induced from d) as ranking criterion, setting $\mu = 2000$ (as for all our algorithms). (Note

[7] Recall from Section 2.3 (page 13) that this is equivalent to ranking by query likelihood.

that this ranking criterion is equivalent to using the cross entropy — details can be found in Section 2.3). The free parameters are the number of highest-ranked documents ($|\mathcal{D}_{\text{init}}|$) used for estimation and the interpolation parameter λ, which controls the evaluation of the highest-ranked-documents' language models.

Following past work on relevance models (Connell et al., 2004; Cronen-Townsend et al., 2004; Metzler et al., 2005), and our implementation of Rocchio's model, we also experimented with *clipping* the relevance model to assign non-zero probability to only a restricted number of terms (up to a maximum of several hundred) — those with the highest $p_{\mathcal{R}}(\cdot)$ value.

Although our reference comparison models operate in different spaces (vector space vs. the probability simplex), as can be seen in Equation 4.3 and as Lavrenko and Croft (2003) observe (with regard to the i.i.d sampling we employed here), both the pseudo-feedback version of Rocchio and the relevance model utilize a linear combination of the top-retrieved documents' models to construct an *expanded query model.*

The number of top retrieved documents ($|\mathcal{D}_{\text{init}}|$) used for the estimation of both pseudo-feedback methods (Rocchio's model and the relevance model) was chosen from $\{5, 10, 20, \ldots, 100\}$. The number of terms used for expansion in Rocchio's method and the clipped relevance model was set to a value in $\{5, 10, \ldots, 50, 100, \ldots, 600\}$. (As mentioned above, we also experimented with Rocchio's original model without employing term clipping, and the performance results we present reflect the better performing model of Rocchio's — i.e., with or without clipping.)

The interpolation parameter λ for the (clipped) relevance model and the weighting coefficient for Rocchio's method was set to a value in $\{0.1, \ldots, 0.9\}$.

Table 4.4 presents the performance numbers for the interpolation-t algorithm and the pseudo-feedback methods. Our first observation is that the performance of the interpolation-t algorithm is better than that of Rocchio's method in 4 out of the 6 relevant comparisons (2 evaluation measures × 3 corpora), and in the other two relevant comparisons, the performance differences are neither substantial nor statistically significant. Furthermore, on LA+FR the interpolation-t algorithm posts substantial and statistically significant performance improvements over Rocchio's model.

We can also see in Table 4.4 that our interpolation-t algorithm performance is in most cases better than that of the non-clipped relevance model; on LA+FR the differences are not only substantial but also statistically significant.

When comparing the interpolation-t algorithm with the clipped relevance model (Table 4.4), we observe that the latter posts small (non statistically-significant) performance improvements over the interpolation-t algorithm on AP89, but clearly has an inferior performance on LA+FR (though not a to a statistically significant degree). However, on AP88+89, the clipped relevance model has statistically significantly better performance than that of the interpolation-t algorithm. We attribute the latter differences to the extensive optimization employed for the number of terms used for expansion for the clipped relevance model. Indeed, the differ-

Table 4.4: Comparison of the interpolation-t algorithm with Rocchio's method, the relevance model and the clipped relevance model. (Boldface: best result in column.) Statistically significant differences with the three methods are marked with η, $\mathcal{R}$, and c respectively. No stemming or stopword removal have been applied.

	AP89		AP88+89		LA+FR	
	prec	recall	prec	recall	prec	recall
Rocchio	22.62%	60.9%	28.91%	74.71%	17.88%	51.4%
Rel. Model	23%	50.29%	30%	71.8	22.03%	46.59%
Clipped Rel. Model	**24.49%**	**61.67%**	**31.84%**	**80.5%**	22.34%	56.65%
interpolation-t	24.16%	60.20%$^{\mathcal{R}}$	28.45%^{c}	73.88%^{c}	**24.05%**$^{\eta\mathcal{R}}$	**64.20%**$^{\eta\mathcal{R}}$

ences between the interpolation-t algorithm performance and that of the relevance model are not significant on AP88+89. (In fact, the latter had worse recall performance than that of the interpolation-t algorithm on AP88+89.) Furthermore, term clipping might have also helped to improve the performance of the interpolation-t algorithm, if employed on the clusters' language models, for example, by improving the estimates for cluster-query and document-cluster associations.

Our conclusion in light of the above is that with moderate optimization, our interpolation-t algorithm has performance that in general is competitive with that of highly optimized state-of-the-art pseudo-feedback methods. While on AP88+89 some of the reference comparisons post improvements over the interpolation-t algorithm, on AP89 the performance of the latter is in some cases better than that of the pseudo-feedback methods, and on the LA+FR corpus, which is the most heterogeneous corpus among the three corpora tested, the interpolation-t algorithm has performance that is substantially (and in some cases statistically significant) better than that of the pseudo-feedback methods.

4.4.5 Cluster composition: alternatives and analysis

Cluster-based information plays a crucial role in our algorithms. Indeed, as shown, our algorithms' performance is superior to that of a basic LM approach. This leads us now to examine more closely the nature of the clusters we create. Recall that each of our clusters is the *cohort* of some document d, defined as those documents whose induced language models have smallest KL divergence to the language model induced from d.

How important is it to our algorithms that our clusters overlap? To study this issue, we looked at the effect of using a set of non-overlapping clusters created via the well-known *k-means* algorithm with cosine as document similarity metric; such clusters were used by Liu and Croft (2004) for query-independent clustering in their work on cluster-based language models for ad hoc retrieval. (Since the KL divergence is asymmetric, it is not well-suited to serving as a basis for measuring similarity for k-means.) Specifically, we ran the interpolation-t algorithm on k-means clusters rather than our nearest-neighbor clusters. Note that doing so

Table 4.5: Average precision of the interpolation-t algorithm using non-overlapping k-means clusters (cosine similarity metric) versus overlapping nearest-neighbor clusters (KL divergence-based similarity metric). Bold indicates the best performance for a given experimental setting (column). Stars (*) indicate statistically significant differences with respect to the baseline.

	MED	CISI	CACM
LM (baseline)	44.79%	12.95%	25.31%
interpolation-t (K-MEANS)	*46.90%*	*13.52%*	*25.53%*
interpolation-t (NN)	**63.60%***	**17.72%***	**28.71%***

reduces the interpolation-t algorithm's score function to

$$\delta\big[c \in \text{TopClusters}_q(m)\big] \cdot (\lambda p_d(q) + (1-\lambda)p_c(q)),$$

where c is the sole cluster that contains the document d being scored. If one then sets m, the number of top clusters retrieved, to its maximum possible value, then the above function becomes

$$\lambda p_d(q) + (1-\lambda)p_c(q),$$

which is the *CBDM model* proposed by Liu and Croft(2004) (modulo details regarding how to estimate the language models themselves), as mentioned in Section 4.2. We used smaller corpora than those utilized in the experiments reported above to allow for clustering the corpora numerous times with varying values of k (we searched for the values of k optimizing average precision); in particular, we worked with MED (1033 documents, best $k = 250$), CISI (1460 documents, best $k = 300$), and CACM (3204 documents, best $k = 200$).

Table 4.5 shows the change in the average precision results for the interpolation-t algorithm when we use k-means clusters. As can be seen, the use of non-overlapping clusters yields results that are statistically indistinguishable from those of the baseline on all three corpora. In contrast, use of the language-model-based, overlapping cohorts as clusters allows the interpolation-t algorithm to significantly outperform the baseline.

While the results above indicate that hard partitioning of the corpus results in performance substantially inferior to that of using overlapping clusters, we are also interested in examining the question of whether the large amount of overlapping induced by our nearest-neighbor clustering is indeed crucial for the success of our algorithms. (Recall that the number of clusters we create is the same as the number of documents in the corpus.)

To decrease the amount of overlapping, we present two heuristics for producing a reduced number of nearest-neighbor clusters that are cohorts of documents in the corpus. We maintain the invariant that every document is contained in at least one cluster. (Otherwise some documents will never be assigned a score by our definition of $\text{Facets}_q(d)$ in Table 4.1.) Since the clusters we create are cohorts of documents, the resultant set of clusters is a subset of the original set $Cl(\mathcal{C})$.

Our heuristics, *h-basis* and *h-all*, are based on a greedy algorithm that traverses the documents in the corpus in some order, say $d_1, \cdots, d_{|\mathcal{C}|}$, and that decides whether to use the next document d_i in the list as a basis for adding $\text{Cohort}(d_i)$ to the list of clusters according to the following criteria:

Heuristic	Use d_i as basis for creating $\text{Cohort}(d_i)$ iff
h-basis	$\nexists d_j$ s.t. $j < i$ and $d_i \in \text{Cohort}(d_j)$
h-all	$\exists d \in \text{Cohort}(d_i)$ s.t. $d \notin \bigcup_{j=1}^{i-1} \text{Cohort}(d_j)$

In other words, by the h-basis heuristic a cluster in the original set of clusters $Cl(\mathcal{C})$ is filtered out if its basis document is a member of a previous cluster (wherein the ordering of clusters corresponds to the ordering of their respective basis documents). According to h-all, a cluster is filtered out only if all its constituent documents are members of previous clusters.

To experiment with the reduced set of clusters, we randomize the order of documents in the corpus three times, and present the average resultant performance of the interpolation-t algorithm in Table 4.6.

Our first observation from Table 4.6 is that both heuristics substantially reduce the number of clusters[8]. For example, using the h-basis heuristic on LA+FR results in roughly a 33% decrease in the number of clusters (from 183283 to 57068 on average). Naturally, though, the h-basis heuristic results in fewer clusters than those resulting from the application of the h-all heuristic.

Another observation from Table 4.6 is that the application of the heuristics results in performance that is inferior to that obtained using the original list of clusters (as the boldfaced numbers imply) — in many of the randomized runs significantly so. Furthermore, we observe that the application of the h-all heuristic results in a slightly better performance than that obtained using the h-basis heuristics, providing further support to the finding that a reduced number of clusters results in somewhat worse performance.

However, it is clear that for both heuristics employed the resultant performance is still better than that of the basic LM approach. In fact, for both heuristics and for all evaluation settings (3 corpora × 2 evaluation metrics), the resultant performance is significantly better than that of the baseline in *all* the randomized runs (the average results of which are the numbers presented in Table 4.6).

Our conclusion is that while the high degree of overlap between our created clusters seems to be an important factor in the performance of the interpolation-t algorithm, promising results can still be obtained by reducing this overlapping. We hasten to point, however, that further research is required for finding a smaller

[8] The variance in the number of resultant clusters for both heuristics is very small. Furthermore, the variance of the performance results is also very small.

Table 4.6: Effect of applying the h-basis and h-all heuristics on the interpolation-t algorithm's performance. No stemming or stopword removal have been applied. “$h\text{-}x(n_1, n_2, n_3)$” indicates the (rounded average) number of clusters for AP89, AP88+89 and LA+FR respectively that results from applying the “h-x” heuristic. “original” denotes that no heuristic has been applied (i.e., the original set of clusters $Cl(\mathcal{C})$ was used.) Boldface: best result in column.

	AP89		AP88+89		LA+FR	
	prec	recall	prec	recall	prec	recall
LM (baseline)	19.52%	44.89%	21.03%	55.34%	21.72%	48.81%
original (84678, 164597, 183283)	**24.16%**	**60.20%**	**28.45%**	**73.88%**	**24.05%**	**64.20%**
h-basis(10207, 32493, 57068)	22.06%	55.22%	25.83%	69.42%	23.49%	60.48%
h-all(13069, 39200, 64813)	22.66%	56.08%	26.06%	70.74%	23.65%	61.23%

set of cohorts that will result in strong performance without necessitating the computation of nearest neighbors of every document in the corpus, as is the case for the above suggested heuristics, and more generally, for all our algorithms.

Similarity and relevance Now that we have established that our particular choice of clusters seems well justified from a performance point of view, we now consider another question of interest: is there a relationship between documents having similar induced language models and documents having similar relevance status?

One way to answer this question is to compute the extent to which the top clusters with respect to a query tend to contain documents that are relevant to that query. Figure 4.6 shows that indeed, clusters that are poor “matches” to the query (that is, whose language models assign low probability to it) tend to contain a lower percentage of relevant documents. We also see substantial variation from corpus to corpus in the percentage of relevant documents that the top 50 clusters contain: for LA+FR, only about 1 out of the 10 documents in each such cluster tends to be relevant, whereas for AP88+89, on average more than 5 out of 20 documents in each such cluster are relevant.

Another way to study the relationship between language-model-based similarity and relevance is to look at whether the nearest neighbors (in KL-divergence terms) of a relevant document tend to also be relevant or not (cf. Voorhees' (1985) nearest-neighbors test for the cluster hypothesis). Figure 4.7 shows that no matter what the corpus, at least 50% of the relevant documents have at least one other relevant document within a 10-document radius, but less than 40% have at least 5 out the 10 nearest neighbors being relevant. For AP89 and AP88+89, roughly 8% of the relevant documents have the “ideal” property of having all 10 of their nearest neighbors also being relevant.

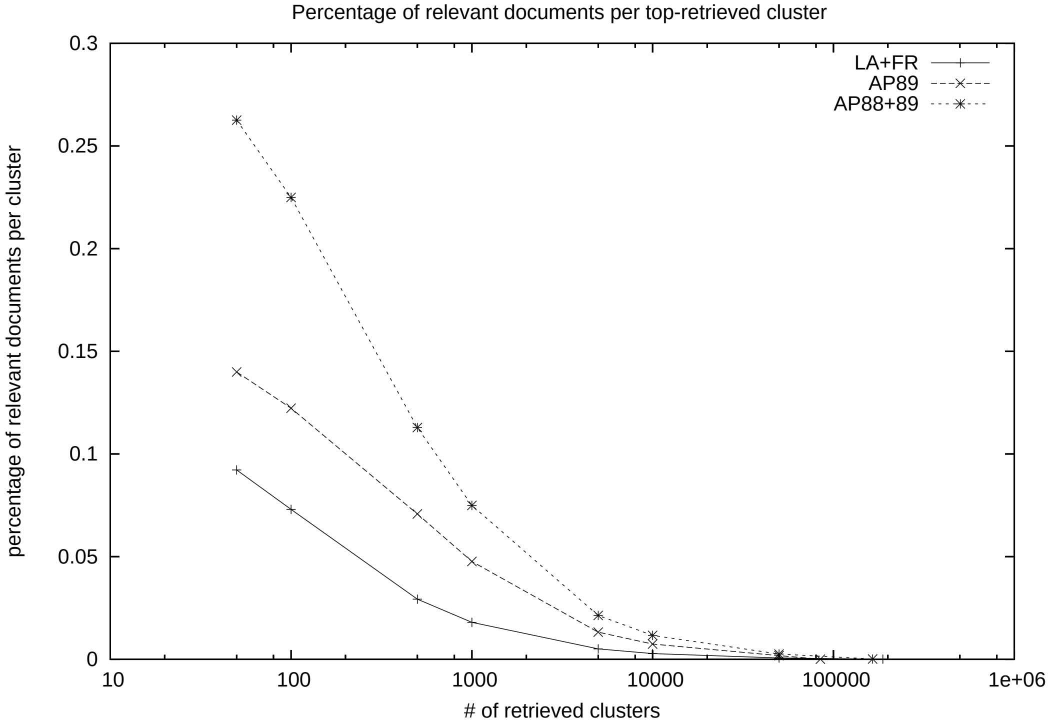


Figure 4.6: Average percentage of relevant documents per top-retrieved cluster as the number of clusters retrieved grows.

4.4.6 The role of language models

Throughout this chapter, we have considered a language-model-based framework for incorporating cluster information into the ranking decision. However, if we think of our estimate of $p_x(y)$ as being one way (that happens to be asymmetric) of measuring the similarity between two text items x and y, then we can view the algorithm template we have proposed (see Figure 4.1) as specifying a subset of a broader family of algorithms, wherein $p_x(y)$ can be replaced by some other measure of similarity between the two items. Doing so allows us to study the specific contribution of the language models themselves, as opposed to the general effect of integrating cluster and document information.

We therefore experimented with replacing all quantities of the form $p_x(y)$ with the inner product between the tf.idf vectors[9] formed from x and y (as before, clusters were treated as large documents formed by concatenating their constituent documents). Altering our selection algorithms (basis-select, set-select, and bag-select) in this way leads to improved performance with respect to the basic tf.idf retrieval algorithm, as shown in Table 4.7. On the other hand, these algorithms did not do as well as their original, LM-based counterparts (in most of the relevant comparisons, especially with respect to average precision), as can be seen by

[9] We used $tf \cdot \log(idf)$ weights in the vectors.

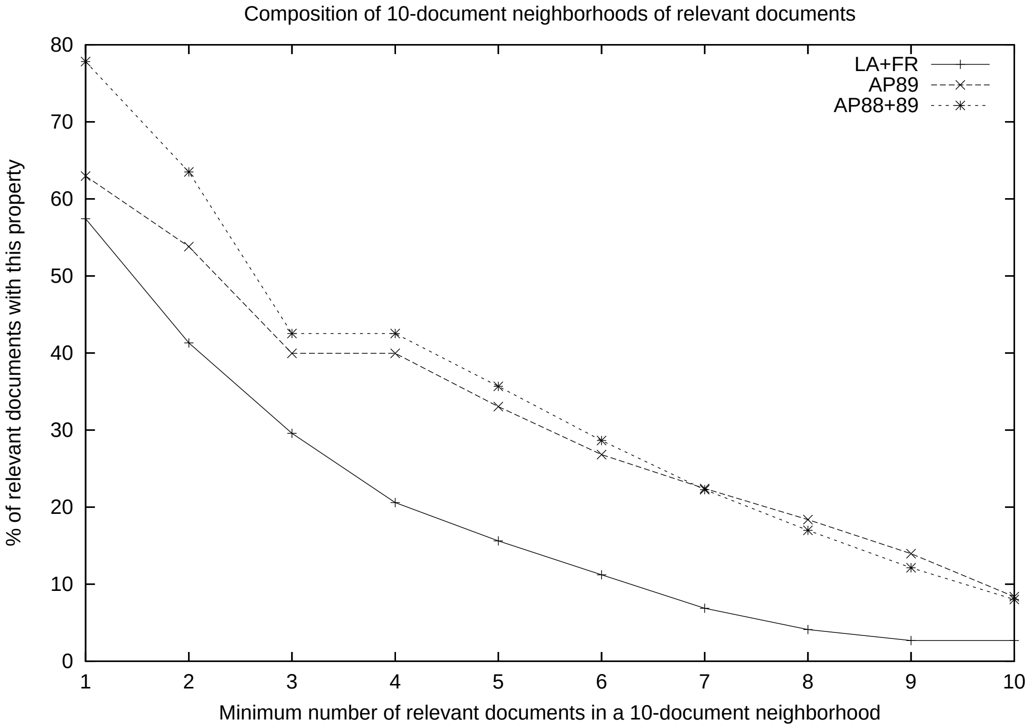


Figure 4.7: For each number x, the percentage of relevant documents r such that at least x of the 10 nearest-neighbors documents to r (not including r itself, and where similarity is calculated based on the documents' induced language models) are relevant.

comparison of Table 4.7 with our primary-evaluation results given in Figure 4.2. We thus see that our general cluster-based algorithmic framework can boost performance over document-specific retrieval techniques when alternative similarity measures are applied, but language models do seem to have some advantages, at least in comparison to our specific choice for vector-space-model representation.

Table 4.7: Results using tf.idf to measure similarity instead of probabilities assigned by induced language models. Italics indicate results superior to those of the baseline tf.idf-based ranking. A "b" marks a statistically significant difference from the baseline. Bold highlights the best performance for a given experimental setting (column). No stemming or stop-word removal was applied.

	AP89		AP88+89		LA+FR	
	prec	recall	prec	recall	prec	recall
TF.IDF	18.01%	45.38%	19.04%	55.46%	16.29%	47.66%
TF.IDF-basis-select	*19.32%*	*52.44%*	*23.71%*	*65.58%*	*16.68%*	*55.93%*
TF.IDF-set-select	*19.01%*^{b}	*52.74%*^{b}	*24.25%*^{b}	*71.80%*^{b}	**16.73%**^{b}	*52.19%*
TF.IDF-bag-select	**22.32%**^{b}	**56.82%**^{b}	**25.79%**^{b}	**73.71%**^{b}	15.17%	**56.00%**^{b}

Chapter 5
Structural re-ranking utilizing query-dependent cluster-based language models

In the previous chapter we presented an algorithmic framework for ad hoc retrieval that exploits language models induced from clusters created offline. As mentioned, one of the challenges in utilizing clusters that are created in a query-independent fashion is to relate the information they provide to the information need specified in a user's query. In this chapter, we study the effectiveness of our cluster-based algorithms when implemented with *query-dependent clusters*, that is, clusters created from an initial list of documents retrieved in response to a query. We study the resultant retrieval algorithms in the context of a *re-ranking* setting: re-ordering an initially retrieved list of documents to obtain high precision at top ranks.

5.1 Structural re-ranking

High precision at the top ranks of the list of retrieved results is highly important for users of search engines (Croft, 1995). Naturally, users want to find the documents pertaining to their information needs as quickly as possible, and avoid traversing long lists in an attempt to locate potentially relevant documents. The ability of a retrieval method to position relevant documents at the very highest ranks of the returned results is sometimes termed *high accuracy retrieval* (Allan, 2003; Shah and Croft, 2004).

Furthermore, high precision at top ranks is also important for applications that utilize ad hoc retrieval as an intermediate step. The implementation of *question answering* systems (Voorhees, 2002), for example, is often based on an initial retrieval step wherein the top retrieved documents are assumed to contain the information required to answer the question at hand. Naturally, high precision at the top ranks of the initial list is very important for the overall success of the system (Collins-Thompson et al., 2004).

An approach that some researchers have taken with the aim of achieving high accuracy retrieval is that of *structural re-ranking*: re-ranking an initial list of documents retrieved in response to a query based on inter-document similarities within the list[1]. As noted throughout the first chapters of this thesis, clusters are convenient means for representing inter-document similarities. Indeed, the potential merits of using *query-dependent* clusters, that is, clusters created from documents in the initial list, for re-ranking have long been recognized (Preece, 1973).

[1] There are also some re-ranking approaches that do not use inter-document similarities within the initial list, but rather use, for example, term co-occurrence information within the list (Mitra et al., 1998).

In one of the earliest works on fully automatic retrieval with query-dependent clusters, Willett (1985) showed that ranking such clusters based on their similarity to the query resulted in a slightly worse performance than that of an equivalent ranking approach that utilized *static clusters* (i.e., clusters created offline). However, as Willett states (1985), this could be attributed to the fact that correlation-based ranking was employed rather than a more effective ranking approach.

Further research on query-dependent clusters demonstrated their potential merits for retrieved-results visualization and navigation (Hearst and Pedersen, 1996; Leuski, 2001). The main benefits in using query-dependent clusters for such tasks are that relevant documents in the initial list "tend" to cluster together — therefore supporting van Rijsbergen's cluster hypothesis (1979) in the re-ranking setting — and clusters can therefore enable a user to quickly detect relevant documents.

One interesting open question posed by both Hearst and Pedersen (1996) and Tombros et al. (2002) was *optimal cluster detection*. They showed that if the initial list is clustered, then there is always a cluster that, if retrieved in its entirety, could be used to achieve performance better than that obtained by document-based retrieval performed over the entire corpus. Moreover, not only was this result true for both a partitioning of the list (Hearst and Pedersen, 1996) and for agglomerative clustering applied to it (Tombros et al., 2002), but it also held for different types of comparisons to document-based retrieval (Tombros et al., 2002). However, automatically finding the optimal cluster — the one whose retrieval would result in the best performance — remains a hard challenge. (We present some progress on this front in Chapter 7.)

Recently, in the language modeling framework, Liu and Croft (2004) showed that clustering the initial list with an agglomerative clustering technique, and smoothing a document language model with the language model of the single cluster to which it belongs (for use in query-likelihood scoring), results in ranking that is somewhat better than the initial ranking of the list.

Throughout the rest of this chapter we examine the performance of (some of) the cluster-based algorithms presented in Chapter 4 when adapted to the re-ranking setting. Through an array of experiments we examine the effect of factors such as different clustering schemes and the set of clusters to be used for smoothing in our aspect-t and interpolation-t algorithms.

It is also important to observe here that one disadvantage of using query-dependent clustering is the computational cost involved in creating the clusters. In contrast to offline clustering, wherein the clusters are created once and then used for all queries, with query-dependent clustering each query requires a new clustering to be performed upon the (changed) list of retrieved documents. Therefore, several researchers proposed fast clustering algorithms for clustering retrieved results (Cutting et al., 1992; Zamir and Etzioni, 1998). The focus of the work in this chapter (and in Chapter 7), on the other hand, is on the potential effectiveness in exploiting clustering and not the efficiency of the clustering method. In fact, as we show in Section 5.3.5, our best performing algorithm has very good performance for several different clustering methods.

5.2 Retrieval framework

Since we are focused on the re-ranking setting, our algorithms are applied not to the entire corpus $\mathcal{C}$, but to a subset $\mathcal{D}_{\text{init}}^{N;q}$ (henceforth $\mathcal{D}_{\text{init}}$), defined as the top N documents retrieved in response to the query q by a given initial retrieval engine. Our algorithms also take into account a set $Cl(\mathcal{D}_{\text{init}})$ of clusters of the documents in $\mathcal{D}_{\text{init}}$.

We now adapt the algorithmic framework from Figure 4.1 (page 28), originally designed for utilizing query-independent clusters, to the re-ranking setting. The framework is presented in Figure 5.1.

```
Create $Cl(\mathcal{D}_{\text{init}})$
For each $d \in \mathcal{D}_{\text{init}}$,
    Choose a cluster subset $\text{Facets}_q(d) \subseteq Cl(\mathcal{D}_{\text{init}})$
    Score $d$ by a weighted combination of $p_d(q)$ and
        the $p_c(q)$'s for all $c \in \text{Facets}_q(d)$
Use the obtained scores for ordering the documents in $\mathcal{D}_{\text{init}}$
```

Figure 5.1: Algorithm template for re-ranking an initially retrieved list of documents ($\mathcal{D}_{\text{init}}$). This template is an adaptation of the template from Figure 4.1 to the re-ranking setting.

As can be seen in Figure 5.1, we do not employ the optional step of re-ranking by $p_d(q)$ that was presented in the original framework (Figure 4.1), since the initial list $\mathcal{D}_{\text{init}}$ was obtained by a retrieval performed in response to q, and therefore such a re-ranking step would potentially restore the initial ranking. (In fact, in our experimental setup (see Section 5.3), such a re-ranking step will result in a ranking equivalent to the initial ranking of the list, since we use a language-model-based approach to create the initial list.)

Clustering algorithm We use the nearest-neighbor clustering approach from Chapter 4 to cluster $\mathcal{D}_{\text{init}}$ into the set of clusters $Cl(\mathcal{D}_{\text{init}})$. Each document $d \in \mathcal{D}_{\text{init}}$ forms the *basis* of a cluster Cohort(d) consisting of d and its $k-1$ nearest neighbors (from $\mathcal{D}_{\text{init}}$) in the language-model space. In Section 5.3.2 we study the performance of our best performing algorithm with alternative clustering schemes.

Retrieval algorithms In Table 5.1 we present adaptations of some of the algorithms from Chapter 4 to the re-ranking setting, as instantiations of the template in Figure 5.1. As noted above, one major difference from the original setting for which these algorithms were designed (i.e., ranking all documents in a corpus using clusters created offline) is the omission of the additional re-ranking step. Another important difference is the definition of the set of clusters $\text{Facets}_q(d)$. Not only is it now a subset of $Cl(\mathcal{D}_{\text{init}})$ — clusters containing only documents from $\mathcal{D}_{\text{init}}$ — but it

Table 5.1: Re-ranking algorithms.

	$\text{Facets}_q(d) \subseteq Cl(\mathcal{D}_{\text{init}})$	Score
basis-select	$\{\text{Cohort}(d)\}$	$p_{\{\text{Cohort}(d)\}}(q)$
bag-select	$\{c : d \in c\}$	$p_d(q) \cdot \lvert\text{Facets}_q(d)\rvert$
aspect-t	$\{c : d \in c\}$	$\sum_{c \in \text{Facets}_q(d)} p_c(q) \cdot p_d(c)$
interpolation-t	$\{c : d \in c\}$	$\lambda \cdot p_d(q) + (1-\lambda) \sum_{c \in \text{Facets}_q(d)} p_c(q) \cdot p_d(c)$

also does not encapsulate the notion of "top-retrieved clusters" ($\text{TopClusters}_q(m)$) as was the case in Table 4.1 (page 28). Indeed, while in Chapter 4 top-retrieved clusters were used to ensure that only clusters that "resemble" the query to a certain extent will be considered at retrieval time, here the clusters contain documents retrieved in response to the query and therefore employing the concept of "top-retrieved clusters" is redundant.

We also observe in Table 5.1 that the basis-select algorithm scores a document by the probability that the language model induced from its cohort assigns to the query, and not by the probability assigned by the language model of the document itself, as was the case in Chapter 4. The performance of this ranking criterion can indicate whether smoothing of a document language model using a nearest-neighbor approach is effective in a re-ranking setting as it is for ranking all documents in a corpus (Tao et al., 2006). Furthermore, using the scoring function from Chapter 4 would have been equivalent to re-ranking $\mathcal{D}_{\text{init}}$ by a basic LM method. As noted above, this would have resulted in no re-ranking since a basic LM approach is used (in our experiments) to create $\mathcal{D}_{\text{init}}$.

5.3 Evaluation

5.3.1 Language model induction

We use the estimate $p_x^{KL,\mu}(\cdot)$ from Equation 2.5 (page 13) for estimating the language-model probabilities $p_d(\cdot)$ and $p_c(\cdot)$ on which the algorithms in Table 5.1 are based. [2] Recall from Section 2.3 that for text x and term sequence $\vec{t}$, the estimate $p_x^{KL,\mu}(\vec{t})$ is equivalent to $\exp\left(H(\vec{t})\right) \cdot p_x^{[\mu]}(\vec{t})^{\frac{1}{|\vec{t}|}}$.

Now, the entropy of a text span can be considered as an indicator of its relevance in the re-ranking setting, as will be shown (for documents) in Chapter 6. However, we hasten to point out that experiments for this chapter utilizing the estimate $p_x^{CE,\mu}(\cdot)$ — which is equivalent to $p_x^{[\mu]}(\vec{t})^{\frac{1}{|\vec{t}|}}$ and therefore does not encapsulate the entropy effect — resulted in performance almost identical to that obtained using $p_x^{KL,\mu}(\cdot)$. To maintain a uniform experimental setup throughout the chapters

[2] We treat a cluster as the concatenation of its constituent documents for the purpose of language model induction (see Section 2.3 in Chapter 2).

discussing the re-ranking setting (Chapters 5, 6 and 7), all experimental results in these chapters utilize the $p_x^{KL,\mu}(\cdot)$ estimate.

We also recall that in Section 4.4.1 the estimate for document-cluster association ($p_d(c)$) was re-scaled to ensure that $\sum_{c:c \ni d} p(c|d) = 1$. Experiments with this re-scaling for the aspect-t and interpolation-t algorithms in this chapter revealed that, in general, the resultant performance was slightly worse than that obtained without re-scaling. Therefore, all experimental results reported in this chapter are based on the above "raw" estimate $p_x^{KL,\mu}(\cdot)$.

5.3.2 Experimental setup

We conducted our experiments on three TREC datasets (Voorhees and Harman, 2000):

corpus	# of docs	queries
AP	242,918	51-64, 66-150
TREC8	528,155	401-450
WSJ	173,252	151-200

We applied basic tokenization and Porter stemming (Porter, 1980) via the Lemur toolkit (www.lemurproject.org), which we also used for language-model induction. Topic titles served as queries.

As described in Section 1.2, we apply here evaluation metrics appropriate to the structural re-ranking task: precision at the top 5 and 10 documents (henceforth prec@5 and prec@10, respectively) and the mean reciprocal rank (MRR) of the first relevant document (Shah and Croft, 2004). All performance numbers are averaged over the set of queries for a given corpus.

We are interested in the general validity of the algorithms presented in Section 5.2 for structural re-ranking. We believe that a good way to emphasize the effectiveness (or lack thereof) of the underlying principles is to downplay the role of parameter tuning. Therefore, we made the following design decisions, with the effect that *the performance numbers we report are purposely not necessarily the best achievable by exhaustive parameter search*:

- The *initial ranking* that created the set $\mathcal{D}_{\text{init}}$ was built according to the estimate $p_d^{KL,\mu}(q)$ where the value of μ was chosen to optimize the non-interpolated average precision of the top 1000 retrieved documents. This is *not* one of our evaluation metrics, but is a reasonable general-purpose optimization criterion. (In fact, results with this initial ranking turned out to be statistically indistinguishable from the results obtained by optimizing with respect to the actual evaluation metrics, although of course they were lower in absolute terms.) In our experiments to follow, we set $\mathcal{D}_{\text{init}}$ to be the 50 highest-ranked documents using the above criterion.

- The value μ in the estimate $p_x^{KL,\mu}(\cdot)$ was set to 2000 in all our re-ranking algorithms, following a general recommendation in Zhai and Lafferty (2001b).

Table 5.2: Experimental results, showing algorithm performance with respect to our 9 evaluation settings (3 performance metrics × 3 corpora). For each evaluation setting, improvements over the optimized baselines are given in *italics*; statistically significant differences between the structural re-ranking algorithms and the initial ranking and optimized baselines are indicated by i and o respectively; **bold** highlights the best results over all algorithms.

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
upper bound	.876	.788	.930	.944	.850	.980	.896	.800	1.000
init. ranking	.457	.432	.596	.500	.456	.691	.536	.484	.748
opt. baselines	.465	.439	.635	.512	.464	.696	.560	.494	**.772**
basis-select	.438	.413 $_{o}$	.554 $_{o}$	.468	.460	.591 $^{i}_{o}$	.456 $^{i}_{o}$	.450	.636 $^{i}_{o}$
bag-select	*.507*	*.494* $^{i}_{o}$	.630	*.532*	**.514** $^{i}_{o}$	.660	.548	.488	.719
aspect-t	*.517* ^{i}	*.496* $^{i}_{o}$	**.654**	*.548*	*.484*	.688	.528	**.496**	.689
interpolation-t	**.527** $^{i}_{o}$	**.499** $^{i}_{o}$	*.651*	**.564** ^{i}	*.494*	**.707**	**.568**	.490	.725

- We only optimized settings for k (cluster size) and λ (the interpolation parameter in the interpolation-t algorithm) with respect to precision among the top 5 documents[3], not with respect to all three evaluation metrics employed. As a consequence, our prec@10 and MRR results are presumably not as high as possible; but the advantage of our policy is that we can see whether optimization with respect to a fixed criterion yields good results no matter how "goodness" is measured. (Refer back to Section 1.2 for more details about our evaluation methodology.)

Parameter values were selected from the following sets. The cluster size k: $\{2, 5, 10, 20, 30\}$. The interpolation parameter λ: $\{0.1, 0.2, \ldots 0.9\}$.

In what follows, when we say that results or the difference between results are "significant", we mean according to the two-sided Wilcoxon test at a confidence level of 95%.

A ranking method might assign different documents the same score; we break such ties by item ID. Alternatively, the scores used to determine $\mathcal{D}_{\text{init}}$ can be utilized, if available.

5.3.3 Primary evaluations

Our main experimental results are presented in Table 5.2. The first three rows specify reference-comparison data. The initial ranking was, as described above, produced using $p_d^{KL,\mu}(q)$ with μ chosen to optimize for non-interpolated precision at 1000. The *empirical upper bound on structural re-ranking*, which applies to any

[3] If two different parameter settings yield the same prec@5, we choose the setting *minimizing* prec@10 so as to provide a conservative estimate of expected performance. Similarly, if we have ties for both prec@5 and prec@10, we choose the setting minimizing MRR.

algorithm that re-ranks $\mathcal{D}_{\text{init}}$, indicates the performance that would be achieved if all the relevant documents within the initial fifty were placed at the top of the retrieval list. We also computed an *optimized baseline* for each metric m and test corpus $\mathcal{C}$; this consists of ranking all the documents (not just those in $\mathcal{D}_{\text{init}}$) by $p_d^{KL,\mu}(q)$, with μ chosen to yield the best m-results on $\mathcal{C}$. As a sanity check, we observe that the performance of the initial retrieval method is always below that of the corresponding optimized baseline (though not statistically distinguishable from it).

Our first observation from Table 5.2 is that the interpolation-t algorithm is the best performing method among the ones presented in Table 5.1 for structural re-ranking. In particular, interpolation-t always outperforms the other re-ranking algorithms with respect to prec@5 — the evaluation metric for which we optimized performance. Furthermore, in 7 of the 9 evaluation settings (3 corpora × 3 evaluation metrics), the interpolation-t algorithm outperforms the optimized baselines — sometimes to a significant degree, though applied to a sub-optimally-ranked initial list. The aspect-t algorithm also posts effective performance — in the majority of the cases superior to that of the initial ranking — but not as effective as that of the interpolation-t algorithm. This indicates that although our algorithms operate in the re-ranking setting, similarity to the query is still an important source of information.

Further analysis of the results in Table 5.2 reveals that the basis-select algorithm, as adapted to the re-ranking setting in Table 5.1, is not effective for structural re-ranking, as its performance is always below that of the initial ranking. This result is not surprising, since the basis-select algorithm ranks documents in $\mathcal{D}_{\text{init}}$ by the "match" of their smoothed language models to the query, wherein smoothing is based on nearest neighbors in $\mathcal{D}_{\text{init}}$. Since all documents in $\mathcal{D}_{\text{init}}$ are somewhat similar to one another (by virtue of them being deemed similar to q), nearest-neighbor smoothing makes the differentiation between them even harder.

The bag-select algorithm, on the other hand, is very effective for structural re-ranking. It outperforms the initial ranking in 7 out of the 9 relevant settings (3 corpora × 3 evaluation metrics). Specifically, it always posts improvements over the initial ranking when prec@5 — the evaluation metric for which performance was optimized — is concerned.

When comparing the relative performance patterns of our algorithms as employed for the re-ranking setting to those they exhibited in the setting presented in the previous chapter (see Figure 4.2, page 35), that is, ranking all documents in a corpus in response to a query, we observe that the relative performance patterns are quite consistent except for the aspect-t algorithm posting performance that is often better than that of the bag-select algorithm in the re-ranking case. We believe that this difference stems from the fact that over-generalization, which is a major problem for the aspect-t algorithm in the ranking setting, is not as severe a problem in the re-ranking setting since the clusters contain only documents that bear high similarity to the query.

Table 5.3: Comparison between the "truncated" (-t) and "full" (-f) versions of the aspect and interpolation algorithms. Underline: best result in a "block" (corpus × algorithm × evaluation measure). Boldface: best result per column. Statistically significant differences with the initial ranking and optimized baselines are marked with i and o respectively.

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
init. ranking	.457	.432	.596	.500	.456	.691	.536	.484	.748
opt. baselines	.465	.439	.635	.512	.464	.696	.560	.494	**.772**
aspect-t	$.517^{i}$	$.496^{io}$	$\underline{\mathbf{.654}}$	.548	.484	.688	.528	.496	.689
aspect-f	$\underline{\mathbf{.537}}^{io}$	$\underline{.498}^{io}$	.628	$\underline{.560}^{i}$	$\underline{\mathbf{.504}}^{i}$	$\underline{\mathbf{.714}}$	$\underline{.576}$	$\underline{.504}$	$\underline{.759}$
interpolation-t	$.527^{io}$	$\underline{\mathbf{.499}}^{io}$	$\underline{.651}$	$.564^{i}$	.494	$\underline{.707}$	.568	.490	.725
interpolation-f	$\underline{\mathbf{.537}}^{io}$	$.498^{io}$	.628	$\underline{\mathbf{.576}}^{i}$	$\underline{.496}$	.687	$\underline{\mathbf{.592}}^{i}$	$\underline{\mathbf{.508}}$	$\underline{.767}$

5.3.4 Cluster-document relationship

In both the aspect-t and interpolation-t algorithms, clusters can "represent" (or smooth the language model of) only documents that they contain. (Recall the definition of the $\text{Facets}_q(d)$ set in Table 5.1.) In Section 4.4.5 (Chapter 4) we saw that the high degree of cluster overlap had a considerable effect on the performance of the interpolation-t algorithm, i.e., "representing" documents via multiple clusters seemed to result in improved retrieval performance.

Therefore, we examine here the question of whether "representing" documents in $\mathcal{D}_{\text{init}}$ via more clusters than originally proposed in Table 5.1 can enhance the re-ranking performance of the aspect-t and interpolation-t algorithms.

For this purpose, we define the **aspect-f** and **interpolation-f** algorithms to utilize the same scoring functions of aspect-t and interpolation-t respectively, but to use all the clusters in $Cl(\mathcal{D}_{\text{init}})$ as the facets group (i.e., we set $\text{Facets}_q(d) = Cl(\mathcal{D}_{\text{init}})$). [4] Thus, the aspect-f algorithm scores a document $d \in \mathcal{D}_{\text{init}}$ by:

$$\sum_{c \in Cl(\mathcal{D}_{\text{init}})} p_c(q) p_d(c),$$

(note that this is exactly the aspect model of Hofmann and Puzicha (1998)) and the interpolation-f algorithm, then, scores d by:

$$\lambda p_d(q) + (1 - \lambda) \sum_{c \in Cl(\mathcal{D}_{\text{init}})} p_c(q) p_d(c).$$

(See Section 4.3 (Chapter 4) for the mathematical derivation.).

[4] The suffix "-f" stands for "full", to indicate that the scoring function does not employ any sum truncation, nor is any constraint imposed on the clusters participating in this summation. Recall that in Chapter 4, our experiments with the aspect-t and interpolation-t algorithms also utilized all available clusters (i.e., all clusters in the corpus), but, as opposed to the "-f" algorithms here, a document's facets group contained only clusters to which it belongs.

Table 5.3 presents a comparison of the performance results of the aspect-f and interpolation-f algorithms with those of the original "truncation"-based algorithms (aspect-t and interpolation-t respectively). These results clearly indicate that using all the clusters in $Cl(\mathcal{D}_{\text{init}})$ to "represent" — to a degree determined by $p_d(c)$ — each document instead of just using the clusters to which the document belongs is beneficial for performance. (Note that most of the underlined numbers appear in "-f" rows.)

Furthermore, in terms of prec@5 — the metric for which performance was optimized — the interpolation-f algorithm always improves on the initial ranking by a wide margin that is also statistically significant.

5.3.5 Alternative clustering schemes

Heretofore, we have focused on nearest-neighbor overlapping clusters. However, our aspect and interpolation algorithms can utilize hard clusters as well, whether we use their "truncated" or "full" versions. Indeed, as mentioned in Section 4.4.5, the interpolation-t algorithm implemented with hard clusters is essentially the CBDM model proposed by Liu and Croft (2004), who tested its performance utilizing agglomerative clustering techniques in the re-ranking setting. We recall that the CBDM model scores a document d by:

$$\lambda p_d(q) + (1 - \lambda) p_c(q),$$

where c is the single hard cluster to which d belongs (i.e., $\text{Facets}_q(d) = \{c\}$).

To study the effect of the clustering scheme on the performance of the interpolation algorithm, we now turn to an analysis of its two versions (i.e., interpolation-t and interpolation-f) when implemented with hard clusters, i.e., $\text{Facets}_q(d)$ will now be the set of clusters resulting from clustering $\mathcal{D}_{\text{init}}$ using a hard clustering scheme.

One hard clustering scheme that we employ is *agglomerative clustering*, which was extensively used in previous work on structural re-ranking (Willett, 1985; Leuski, 2001; Tombros et al., 2002; Liu and Croft, 2004). The different criteria we use for merging clusters are: single link, complete link, average distance, centroid distance and the Ward criterion (El-Hamdouchi and Willett, 1986). Descriptions of these criteria, with a focus on the re-ranking setting, can be found in Leuski (2001). To produce a list of (non-overlapping) clusters using these different criteria, we utilize a bottom-up merging approach, and stop the merging process when the number of clusters is as required. The number of clusters we use in our experiments is chosen from the set $\{2, 5, 10, 25\}$ so as to roughly result in an average cluster size equivalent to the cluster size used for the nearest-neighbor clustering method we used so far.

We also use the k-means clustering algorithm, so as to follow previous work on visualization in the re-ranking setting that utilized partitioning algorithms (Cutting et al., 1992; Hearst and Pedersen, 1996). We set k to a value in $\{2, 5, 10, 25\}$ to comply with the choice we made for the agglomerative clustering methods above.[5]

[5] For both the agglomerative clustering and k-means clustering methods, all

The above clustering algorithms require a symmetric similarity measure, and are usually implemented using a vector space representation. We therefore use a "log tf.log idf" representation[6] and the cosine similarity function that was used in previous work on re-ranking using hard clusters (Willett, 1985; Leuski, 2001; Tombros et al., 2002; Liu and Croft, 2004). In addition, for completeness of comparison, we also implement the nearest-neighbor clustering method using the same vector space representation. (We will denote it as "nn-VS" to differentiate it from our original nearest-neighbor-based clustering method which operates in the language-model space and which we denote as "nn-LM").

In Figure 5.2 we present the performance results of the interpolation-t and interpolation-f algorithms when utilizing the different clustering methods. Table 5.4 then summarizes the relative performance patterns of the two algorithms: each entry depicts in non ascending order of performance the algorithms that improve on the initial ranking with at least 2.5% relative difference (a hat (ˆ) indicates that the difference is statistically significant).

Our first observation with respect to Figure 5.2 is that in most of the relevant settings (corpora × evaluation measures), for all clustering methods, the interpolation algorithm (in both its truncated interpolation-t ("-t") and full interpolation-f ("-f") versions) improves on the initial ranking, thereby once again demonstrating its effectiveness as a cluster-based structural re-ranking paradigm.

In comparing the overlapping clustering methods to the hard clustering ones, when using the vector space representation, we see that the interpolation-f algorithm with nn-VS clustering (nearest-neighbors in a vector space) posts the best performance in a majority of the settings. For the interpolation-t algorithm, however, this is not necessarily the case. Nevertheless, we can clearly see that in most of the evaluation settings (for both the interpolation-t and interpolation-f algorithms), the best performing results are obtained for one of the nearest-neighbor clustering methods. Thus, these findings support the hypothesis that overlapping clusters are effective means for modeling structure not only in a query-independent fashion but also in a re-ranking setting.

Comparing the interpolation-t and interpolation-f algorithms, we observe in Table 5.4 that while for the single link, average link and centroid-distance methods the interpolation-t algorithm is superior to the interpolation-f algorithm, for the other five clustering methods the reverse is true. Additional exploration revealed that for the first three (single, average and centroid) the clustering usually resulted in one large cluster along with very few small ones each often containing a single

presented results reflect an optimal choice of the number of clusters as determined by our optimization criterion (see Section 5.3.2).

[6] We note that the weighting scheme for vector-space-model representation can have considerable effect on the resultant performance. Different experiments with estimating inter-document similarities for the work presented in this thesis showed that "log tf.log idf" weights resulted, in general, in better performance than "tf.log idf" weights (Zobel and Moffat, 1998). We further discuss this in Section 6.3.6.

document. Thus, it seems that in these cases the interpolation-t algorithm simply enjoys the separation of what could be considered as “outliers” from the rest of the documents, while the interpolation-f does not, since for each document all clusters are being considered.

All in all, we believe that the most important message arising from this analysis is that the interpolation algorithm is a highly effective paradigm for structural re-ranking that can utilize different clustering methods, whether they result in (soft) overlapping clusters or hard clusters.

Figure 5.2: Performance numbers for the “truncated” (-t) and “full” (-f) versions of the interpolation algorithm (interpolation-t and interpolation-f respectively), utilizing different clustering methods. Boldface: best result per column; i, o: significant difference with the initial ranking and optimized baseline respectively; italics: improvements over the optimized baselines.

(a) Performance of the **interpolation-t** algorithm.

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
init. ranking	.457	.432	.596	.500	.456	.691	.536	.484	.748
opt. baselines	.465	.439	.635	.512	.464	.696	.560	.494	.772
nn-LM	**.527** $^{i}_{o}$	**.499** $^{i}_{o}$	**.651**	**.564** ^{i}	*.494*	*.707*	*.568*	.490	.725
nn-VS	*.519* $^{i}_{o}$	*.474*	*.644*	*.524*	.446	**.748**	*.584*	.492	.761
agg-single	*.521* ^{i}	*.488* $^{i}_{o}$	.620	*.528*	*.490* ^{i}	.662	*.580*	*.530*	**.783**
agg-comp	*.493*	*.467*	.600	*.524*	*.468*	.674	.544	.488	.740
agg-avg	*.525* $^{i}_{o}$	*.490* $^{i}_{o}$	.619	*.540*	**.504** ^{i}	.675	*.572*	**.532** ^{i}	.765
agg-centroid	*.523* $^{i}_{o}$	*.487* $^{i}_{o}$	.592	*.528*	*.492* ^{i}	.662	**.592**	*.530*	*.773*
agg-ward	*.491*	*.465*	.587	.504	.446	.694	*.584*	*.508*	.715
kmeans	*.475*	*.459*	.579	*.532*	*.486* ^{i}	*.720*	*.568*	*.502*	.743

(b) Performance of the **interpolation-f** algorithm.

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
init. ranking	.457	.432	.596	.500	.456	.691	.536	.484	.748
opt. baselines	.465	.439	.635	.512	.464	.696	.560	.494	**.772**
nn-LM	**.537** $^{i}_{o}$	**.498** $^{i}_{o}$	.628	**.576** ^{i}	*.496*	.687	*.592* ^{i}	*.508*	.767
nn-VS	*.523* ^{i}	*.490* ^{i}	**.642**	*.572* ^{i}	*.476*	**.760**	**.608** ^{i}	**.544** $^{i}_{o}$	.758
agg-single	*.493*	*.482* ^{i}	.582	*.516*	*.468*	*.704*	.552	.476	.747
agg-comp	*.513* $^{i}_{o}$	*.480* $^{i}_{o}$	.613	*.564* ^{i}	**.508** ^{i}	*.750*	*.584*	*.534*	.771
agg-avg	*.503*	*.482* $^{i}_{o}$	.599	*.524*	*.468*	*.738*	*.568*	*.506*	.767
agg-centroid	*.493*	*.482* $^{i}_{o}$	.584	.508	*.466*	.673	.556	.494	.748
agg-ward	*.525* $^{i}_{o}$	*.491* $^{i}_{o}$	.612	*.536*	*.484*	.685	*.592*	*.530*	.762
kmeans	*.509*	*.465*	.584	*.556* ^{i}	*.482*	*.701*	.580	*.520*	.753

Table 5.4: Comparison of the interpolation-t (T) algorithm with the interpolation-f (F) algorithm, utilizing different clustering methods. Each entry depicts in non-ascending order of performance the algorithms that post a 2.5% (or more) relative performance improvement over the initial ranking (a hat ("^") indicates that the improvement is significant). Bold highlights the best performing algorithm per entry.

		nn-LM	nn-VS	ag-single	ag-comp.	ag-avg.	ag-cent.	ag-ward	kmeans
AP	prec @5	$\hat{\boldsymbol{F}}\hat{T}$	$\hat{\boldsymbol{F}}\hat{T}$	$\hat{\boldsymbol{T}}F$	$\hat{\boldsymbol{F}}T$	$\hat{\boldsymbol{T}}F$	$\hat{\boldsymbol{T}}F$	$\hat{\boldsymbol{F}}\hat{T}$	$\boldsymbol{F}T$
	prec @10	$\hat{\boldsymbol{T}}\hat{F}$	$\hat{\boldsymbol{F}}T$	$\hat{\boldsymbol{T}}\hat{F}$	$\hat{\boldsymbol{F}}T$	$\hat{\boldsymbol{T}}\hat{F}$	$\hat{\boldsymbol{T}}\hat{F}$	$\hat{\boldsymbol{F}}\hat{T}$	$\boldsymbol{F}T$
	MRR	$\boldsymbol{T}F$	$\boldsymbol{T}F$	$\boldsymbol{T}$	$\boldsymbol{F}$	$\boldsymbol{T}$		$\boldsymbol{F}T$	
TREC8	prec @5	$\hat{\boldsymbol{F}}\hat{T}$	$\hat{\boldsymbol{F}}T$	$\boldsymbol{T}F$	$\hat{\boldsymbol{F}}T$	$\boldsymbol{T}F$	$\boldsymbol{T}$	$\boldsymbol{F}T$	$\hat{\boldsymbol{F}}T$
	prec @10	$\boldsymbol{F}T$	$\boldsymbol{F}$	$\hat{\boldsymbol{T}}F$	$\hat{\boldsymbol{F}}T$	$\hat{\boldsymbol{T}}F$	$\hat{\boldsymbol{T}}$	$\boldsymbol{F}T$	$\hat{\boldsymbol{T}}F$
	MRR		$\boldsymbol{F}T$		$\boldsymbol{F}$	$\boldsymbol{F}$			$\boldsymbol{T}$
WSJ	prec @5	$\hat{\boldsymbol{F}}T$	$\hat{\boldsymbol{F}}T$	$\boldsymbol{T}F$	$\boldsymbol{F}$	$\boldsymbol{T}F$	$\boldsymbol{T}F$	$\boldsymbol{F}T$	$\boldsymbol{F}T$
	prec @10	$\boldsymbol{F}$	$\hat{\boldsymbol{F}}$	$\boldsymbol{T}$	$\boldsymbol{F}$	$\hat{\boldsymbol{T}}F$	$\boldsymbol{T}$	$\boldsymbol{F}T$	$\boldsymbol{F}T$
	MRR	$\boldsymbol{F}$		$\boldsymbol{T}$	$\boldsymbol{F}$	$\boldsymbol{F}$	$\boldsymbol{T}$		

5.3.6 Clusters-mediated similarity vs. distinct document similarity

As mentioned above, the aspect-f algorithm assigns a document d the score $\sum_c p_c(q)p_d(c)$; the interpolation-f algorithm assigns d the score $\lambda p_d(q) + (1 - \lambda)\sum_c p_c(q)p_d(c)$. In these two scoring functions, clusters only play the role of smoothing: their language models are used to smooth d's language model so as to provide "context" within the list $\mathcal{D}_{\text{init}}$.

We therefore ask now the following question: if clusters are indeed required only for providing context within the list $\mathcal{D}_{\text{init}}$, can single documents play the same role? We study this question by simply defining *singleton* clusters, i.e., each document in $\mathcal{D}_{\text{init}}$ serves as a cluster. Then, the scoring functions of the aspect-f and interpolation-f algorithms become $\sum_{d_i \in \mathcal{D}_{\text{init}}} p_{d_i}(q)p_d(d_i)$ and $\lambda p_d(q) + (1 - \lambda)\sum_{d_i \in \mathcal{D}_{\text{init}}} p_{d_i}(q)p_d(d_i)$ respectively.

Note that in this version the aspect-f algorithm assigns relatively high scores to documents that are similar (to a large extent) to many other documents in $\mathcal{D}_{\text{init}}$, and interpolation-f integrates this information with direct similarity to the query.

Table 5.5 presents results using documents as singleton clusters for both the aspect-f and interpolation-f algorithms. Our first observation is that while the aspect-f algorithm (with singleton clusters) only sometimes outperforms the initial ranking, the interpolation-f (with singleton clusters) not only does so in most of the evaluation settings, but also outperforms the optimized baselines in a majority of the cases.

When comparing the resultant performance of the aspect-f and interpolation-f algorithms with nearest-neighbor clusters to that obtained by using singleton clusters (i.e., documents), we clearly see that the former is a much better approach for both algorithms (aspect-f and interpolation-f). This gives further support to

Table 5.5: Comparison of the aspect-f and interpolation-f algorithms performance with nearest-neighbors (in LM space) clusters ("nn-LM") vs. singleton ("single") clusters (i.e., each document serves as a cluster). Underline: best performance within a block (algorithm × corpus × evaluation metric). Boldface: best performance per column; i, o: statistically significant difference with the initial ranking and optimized baseline respectively.

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
init. ranking	.457	.432	.596	.500	.456	.691	.536	.484	.748
opt. baselines	.465	.439	**.635**	.512	.464	.696	.560	.494	**.772**
aspect-f (single)	.499	.477	.622	.508	.448	.653	.520	.484	.687
aspect-f (nn-LM)	$\underline{\textbf{.537}}^{io}$	$\underline{\textbf{.498}}^{io}$	$\underline{.628}$	$\underline{.560}^{i}$	$\underline{\textbf{.504}}^{i}$	$\underline{\textbf{.714}}$	$\underline{.576}$	$\underline{.504}$	$\underline{.759}$
interp.-f (single)	.515	$.497^{io}$	.615	.536	.482	$\underline{.698}$	.564	$\underline{\textbf{.510}}$	.696
interp.-f (nn-LM)	$\underline{\textbf{.537}}^{io}$	$\underline{\textbf{.498}}^{io}$	$\underline{.628}$	$\underline{\textbf{.576}}^{i}$	$\underline{.496}$	.687	$\underline{\textbf{.592}}^{i}$	.508	$\underline{.767}$

the hypothesis that clusters, even if used only for smoothing, play a very important role in our algorithms.[7]

We also observe that the fact that the interpolation-f algorithm almost always outperforms the aspect-f algorithm when singleton clusters are used (as is the case for using nearest-neighbor clusters) implies that while similarity to other documents in the list is by itself a (somewhat) useful source of information, enhancing it with information about similarity to the query can improve performance substantially.

Finally, the resultant retrieval effectiveness of utilizing inter-document similarities in the list $\mathcal{D}_{\text{init}}$, as manifested in the performance of the interpolation-f algorithm when implemented with singleton clusters, will lead us in the next chapter to design a framework that models such similarities as links in a graph and that leverages both information drawn from these links and from similarities to the query.

[7] A similar conclusion with respect to the superiority of clusters to documents as "pseudo-queries" was made in Kurland et al. (2005).

Chapter 6
Language-model-based graph framework for structural re-ranking[1]

In the concluding section of the previous chapter, we saw that pairwise inter-document similarities convey valuable information that can be exploited for structural re-ranking. In the Web setting, for example, explicitly-indicated pairwise relationships — i.e., hyperlinks — help to compute which documents are the most *central.* The PageRank algorithm (Brin and Page, 1998) is one of the most well known examples for computing centrality based on hyperlink information. In this chapter, we consider adapting this idea to the *re-ranking* setting for corpora in which explicit links between documents do not exist. Then, in the next chapter we show how to incorporate cluster-based information into the framework we develop here for improving centrality-based (re-)ranking.

How should we form links in a non-hypertext setting? While previous work in text summarization has applied PageRank to cosine-based links induced for pairs of sentences (Erkan and Radev, 2004), we utilize here statistical language models. Specifically, we employ *generation links*, which are based on the probability assigned by the language model induced from one document to the term sequence comprising another[2]. We thus combine the strengths of two approaches; one is based on language models used both to model textual information *within* documents and to infer links *between* them, and the other induces centrality based on the inferred links.

We note that the analogy between hyperlinks and generation links is not perfect. In particular, one can attribute much of the success of link-based Web-search algorithms to the fact that hyperlinks are (often) human-provided certifications that two pages are truly related (Kleinberg, 1999). In contrast, automatically-induced generation links are surely a noisier source of information. However, since we are focused on a re-ranking setting, the initial set we wish to re-rank ($\mathcal{D}_{\text{init}}$) could be considered as having a reasonable ratio of relevant to non-relevant documents, and thus forms a good foundation for our algorithms.

We also observe that generation links can be thought of as "complementing" hyperlinks, if utilized in a Web setting, rather than replacing (or serving as the means for re-construction of) hyperlinks. Indeed, hyperlinks indicate "connection" between documents that are related by virtue of a decision stemming from various

[1] This chapter is based on work presented in a paper written with Lillian Lee (Kurland and Lee, 2005) that appeared in the proceedings of SIGIR 2005.

[2] While the term "generate" is convenient, we do not think of a "generator" document or language model as literally "creating" others. That is, we do not assume an underlying generative model, in contrast to Lavrenko and Croft (2003), and Lavrenko (2004), *inter alia.* Other work further discusses this issue and proposes alternate terminology (e.g., "render") (Kurland et al., 2005).

author-dependent reasons, while generation links convey “objective” information stemming from textual similarities.

To compute centrality values for a given graph, we propose a number of methods, including variants of PageRank (Brin and Page, 1998) and HITS (a.k.a. hubs and authorities) (Kleinberg, 1998).

Through an array of experiments, we show that centrality, as induced by graph-based methods over our graphs, and relevance are connected. Furthermore, comparisons against numerous baselines show that language-model-based re-ranking using centrality as a form of “document prior” is indeed successful at moving relevant documents in the initial retrieval results higher up in the list.

In further exploration, we demonstrate the merits of our language-model-based link induction method by comparing it with different vector-space-based notions of similarity (including the cosine measure).

Using an additional array of experiments, we show that our centrality measures are superior to measures based on document-specific characteristics, and explore the “clustering” patterns of relevant and non-relevant documents within our generation graphs by studying the patterns of links induced between them.

6.1 Centrality and relevance

In the Web setting, to determine whether a document is a good candidate for answering the information need underlying a user’s query, two types of information (among others) are often considered. The first is the *estimated relevance* — the extent to which a document’s content seems to pertain to a query as judged by the textual similarity between them. The second is the *centrality* of the document, which is estimated based on the graph structure of the Web as induced by the hyperlink structure (Brin and Page, 1998; Kleinberg, 1998).

In this chapter, we adopt the centrality principle to our proposed re-ranking setting wherein documents lack hyperlink information. We follow (in spirit) Kleinberg’s (Kleinberg, 1998) proposal (for the Web setting) to define centrality over an (augmented) initially retrieved list of documents, in trying to couple the notions of relevance and centrality in the (augmented) initial list. However, as just mentioned, in contrast to the Web setting (Kleinberg, 1998), documents in our setting do not have hyperlinks connecting them; furthermore, we focus on the initially retrieved list and do not augment it with additional documents.

Our hypothesis is that a document that is “central” to the initially retrieved list in terms of its being a “good representative” of other documents in the list has good chances of being relevant to the query. Naturally, defining the notions of “centrality” and “good representative” calls for a specific formulation, one example of which we present in the next section. At the very least though, we can assert that a document in the list that contains terms occurring with “high weight” in many other documents in the list has a good chance of being relevant to the query since the documents in the list were retrieved by virtue of their being similar to the query. The concluding section of the previous chapter provided (very) preliminary evidence for the latter observation, as the aspect-f algorithm — with single documents acting as clusters — was shown to be effective for re-ranking on one of our corpora.

6.2 Retrieval framework

Similarly to Section 5.2 we assume throughout this chapter that the following have been fixed: the corpus $\mathcal{C}$ (in which each document has been assigned a unique numerical ID); the query q (composed of a list of terms); and the set $\mathcal{D}_{\text{init}} \subseteq \mathcal{C}$ of most highly ranked documents returned by some initial retrieval algorithm in response to q (this is the set upon which re-ranking is performed). In addition, we assume the value of an *ancestry* parameter α that pertains to our graph construction process has been fixed.

For each document $d \in \mathcal{C}$, $p_d(\cdot)$ denotes a unigram language model induced from d. (Specific estimation details appear in Section 6.3.1). We use g and o to distinguish between a document treated as a "generator" and a document treated as "offspring", that is, something that is generated (details below).

We use the notation (V, wt) for weighted directed graphs: V is the set of vertices and $wt : V \times V \rightarrow \{y \in \Re : y \geq 0\}$ is the *edge-weight function.* Thus, there is a directed edge between every ordered pair of vertices, but wt may assign zero weight to some edges. We write $wt(v_1 \rightarrow v_2)$ to denote the value of wt on edge (v_1, v_2).

6.2.1 Relevance-flow graphs

Our use of language models to form links can be motivated by considering the following two documents:

d_1: Toronto Sheffield Salvador
d_2: Salvador Salvador Salvador

Knowing that d_2 is important (i.e., central or relevant) would provide strong evidence that d_1 is at least somewhat important, because d_2's importance must stem from the term "Salvador", which also appears in d_1. However, knowing that d_1 is very important does *not* allow us to conclude that d_2 is, since the importance of d_1 might stem from its first two terms. Using language models induced from documents enables us to capture this asymmetry in how centrality is propagated: we allow a document d to receive support for centrality status from a document o only to the extent that $p_d(o)$ is relatively large. (If o is not in fact important, the support it provides may not be significant.) Indeed, as shown in Figure 6.1, if for simplicity's sake we induce two *unsmoothed* unigram language models $p_{d_1}(\cdot)$ and $p_{d_2}(\cdot)$ from d_1 and d_2 respectively, we get that $p_{d_1}(d_2) = p_{d_1}(\text{“Salvador”})^3 = (1/3)^3$ is larger than $p_{d_2}(d_1) = 0$ (due to "Sheffield" and "Toronto" not appearing in d_2).

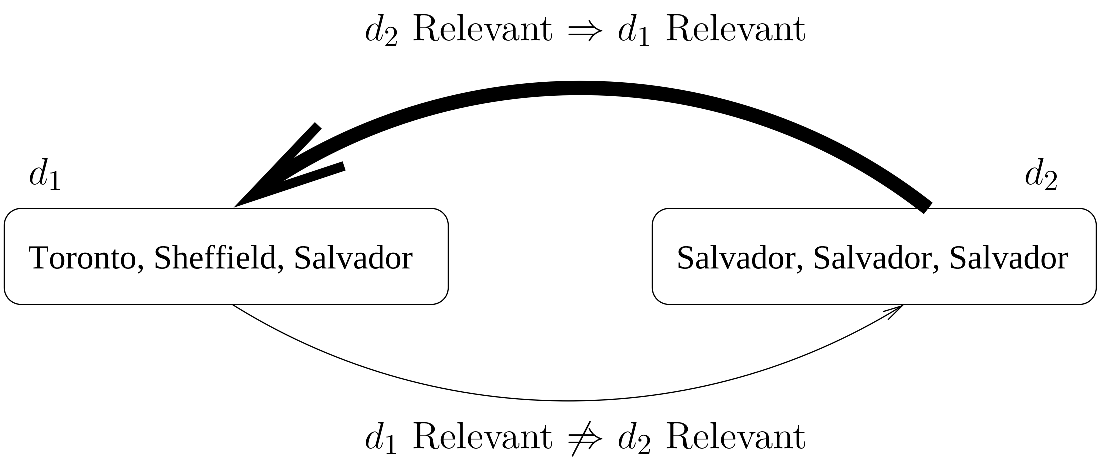

d_2 Relevant $\Rightarrow$ d_1 Relevant
d_1
d_2
Toronto, Sheffield, Salvador
Salvador, Salvador, Salvador
d_1 Relevant $\not\Rightarrow$ d_2 Relevant

Figure 6.1: Intuition behind using language models to induce link information. Assuming unsmoothed unigram language models, $p_{d_1}(d_2) = p_{d_1}(\text{“Salvador”})^3 = (1/3)^3$, which is larger than $p_{d_2}(d_1) = 0$ (due to “Sheffield” and “Toronto” not appearing in d_2). Therefore, the “support” for centrality being transferred from d_2 to d_1 given that d_2 is relevant (thick arrow) is much stronger than the “support” transferred from d_1 to d_2 (thin arrow) given that d_1 is relevant.

Therefore, the link induced between d_2 and d_1 should have higher weight than the opposite link, indicating that d_2 “transfers” more centrality status to d_1 than the other way around[3].

We also note that ranking documents by $p_d(q)$ — i.e., the query likelihood ranking principle (Ponte and Croft, 1998; Miller et al., 1999) — can be considered a variation of our proposed centrality-based principle: given that a query is central (relevant), rank documents by the support for centrality status that they “receive” from the query (as captured by $p_d(q)$).

We are thus led to the following definitions.

Definition 3. *The* top α generators *of a document $d \in \mathcal{D}_{\text{init}}$, denoted $TopGen(d)$, is the set of α documents $g \in \mathcal{D}_{\text{init}} - \{d\}$ that yield the highest $p_g(d)$, where ties are broken by document ID. (We suppress α in our notation for clarity.)*

Definition 4. *The* offspring *of a document $d \in \mathcal{D}_{\text{init}}$ are those documents that d is a top generator of, i.e., the set $\{o \in \mathcal{D}_{\text{init}} : d \in TopGen(o)\}$.*

Note that multiple documents can share offspring, and that it is possible for a document to have no offspring.

[3] The inequality $p_{d_1}(d_2) > p_{d_2}(d_1)$ also holds if smoothed language models are utilized. As noted before, our actual method for language-model induction does involve smoothing (details in section 6.3.1).

We can encode top-generation relationships using either of two *relevance-flow graphs*[4] $G_U = (\mathcal{D}_{\text{init}}, wt_U)$ and $G_W = (\mathcal{D}_{\text{init}}, wt_W)$, where for $o, g \in \mathcal{D}_{\text{init}}$,

$$\begin{aligned} wt_U(o \to g) &= \begin{cases} 1 & \text{if } g \in \mathit{TopGen}(o), \\ 0 & \text{otherwise;} \end{cases} \\ wt_W(o \to g) &= \begin{cases} p_g(o) & \text{if } g \in \mathit{TopGen}(o), \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Thus, in both graphs, positive-weight edges lead only from offspring to their respective top α generators; but G_U treats (edges to) the top generators of o *uniformly*, whereas G_W differentially *weights* them by the probability their induced language models assign to o.

Several of our algorithms (namely, the direct variants of PageRank) rely on the assumption that the graph satisfies certain connectivity properties with respect to those edges with non-zero weight and that for each $o \in \mathcal{D}_{\text{init}}$, $\sum_g wt(o \to g) = 1$ holds. Since G_U and G_W do not satisfy these assumptions, we define "smoothed" versions of them in which all edges (including self-loops) have non-zero weight. To be specific, we employ PageRank's (Brin and Page, 1998) smoothing technique.

Definition 5. *Given an edge-weighted directed graph $G = (\mathcal{D}_{\text{init}}, wt)$ and smoothing parameter $\lambda \in [0, 1)$, the* smoothed *graph $G^{[\lambda]} = (\mathcal{D}_{\text{init}}, wt^{[\lambda]})$ has edge weights defined as follows: for every $o, g \in \mathcal{D}_{\text{init}}$,*

$$wt^{[\lambda]}(o \to g) = \lambda \cdot \frac{1}{|\mathcal{D}_{\text{init}}|} + (1 - \lambda) \cdot \frac{wt(o \to g)}{\sum_{g' \in \mathcal{D}_{\text{init}}} wt(o \to g')}.$$

Note that the definition is valid (i.e., $\sum_{g' \in \mathcal{D}_{\text{init}}} wt(o \to g') \neq 0$), since each document is assigned non-zero generation probability by all documents in $\mathcal{D}_{\text{init}}$ (as a result of applying smoothed language models; details in section 6.3.1). Furthermore, it is easy to see that $\sum_{g' \in \mathcal{D}_{\text{init}}} wt^{[\lambda]}(o \to g') = 1$, and thus the weights of all edges leading out of any given node in $G^{[\lambda]}$ may be treated as *transition probabilities*.

With these concepts in hand, we can now phrase our centrality-determination task as follows: given a relevance-flow graph, compute for each node (i.e., document) how much centrality is "transferred" to it from other nodes — by our edge-weight definitions, centrality therefore corresponds to the degree to which a document is responsible for "generating" (perhaps indirectly) the other documents in the initially retrieved set. We now consider different ways to formalize this notion of transferrence of centrality.

[4] We use the term "relevance-flow" for our graphs as they describe propagation of "centrality status", and under our hypothesis (and as will be shown in Section 6.3.4) centrality and relevance are correlated in the re-ranking setting. The paper upon which this chapter was developed uses the terminology "generation graphs" and so did some work utilizing our graphs to form document representations (Erkan, 2006).

6.2.2 Computing graph centrality

A straightforward way to define the centrality of a document d with respect to a given graph $G = (\mathcal{D}_{\text{init}}, wt)$ is to set it to d's weighted in-degree, which we call its *influx*:

$$Cen_I(d; G) \overset{def}{=} \sum_{o \in \mathcal{D}_{\text{init}}} wt(o \to d). \tag{6.1}$$

The **Uniform Influx** algorithm sets $G = G_U$, so that the only thing that matters is how many offspring d has; it is thus reminiscent of the journal *impact factor* function from bibliometrics (Garfield, 1972), which computes normalized counts of explicit citation links. The **Weighted Influx** algorithm sets $G = G_W$, so that the generation probabilities that d assigns to its offspring are factored in as well.

As previously noted by Pinski and Narin (1976) in their work on *influence weights*, one intuition not accounted for by weighted in-degree methods is that a document with even a great many offspring should not be considered central (or relevant) if those offspring are themselves very non-central. We can easily modify Equation 6.1 to model this intuition; we simply scale the evidence from a particular offspring document by that offspring's centrality, thus arriving at the following *recursive* equation:

$$Cen_{RI}(d; G) \overset{def}{=} \sum_{o \in \mathcal{D}_{\text{init}}} wt(o \to d) \cdot Cen_{RI}(o; G), \tag{6.2}$$

where we also require that $\sum_{d \in \mathcal{D}_{\text{init}}} Cen_{RI}(d; G) = 1$. Unfortunately, for arbitrary G_U and G_W, Equation 6.2 may not have a unique solution or even any solution at all under the normalization constraint just given; however, a unique solution *is* guaranteed to exist for their PageRank-smoothed versions, since in such graphs, the edge weights correspond to the transition probabilities for a Markov chain that is aperiodic and irreducible, and hence has a unique stationary distribution (Grimmett and Stirzaker, 2001) that can be computed by a variety of means (Stewart, 1994; Golub and Van Loan, 1996; Grassmann et al., 1985). [5] In our experiments, power iteration converged very quickly.

By analogy with the two influx algorithms given above, then, we have the **Recursive Uniform Influx** algorithm, which sets $G = G_U^{[\lambda]}$ and is a direct analog of PageRank (Brin and Page, 1998), and the **Recursive Weighted Influx** algorithm, which sets $G = G_W^{[\lambda]}$.

[5] Note that under the original "random surfer" model (Brin and Page, 1998), the sum of the transition probabilities out of nodes with no positive-weight edges emanating from them would be $(1-\lambda)$ not 1 (Ng et al., 2001; Langville and Meyer, 2005). However, by virtue of the edge-weight functions detailed in this chapter, such nodes do not exist in our graphs.

6.2.3 Incorporating initial scores

The centrality scores presented above can be used in isolation as criteria by which to rank the documents in $\mathcal{D}_{\text{init}}$, as our hypothesis states that centrality and relevance should be correlated in the re-ranking setting. However, if available, it might be useful to incorporate more information from the initial retrieval engine to help handle cases where centrality and relevance are not strongly correlated. (Recall that the initial retrieval engine participates in any case by specifying the set $\mathcal{D}_{\text{init}}$.) In our experiments, we explore one concrete instantiation of this approach: we apply language-model-based retrieval (Ponte and Croft, 1998; Croft and Lafferty, 2003) to determine $\mathcal{D}_{\text{init}}$ (as in Section 5.3.2 of Chapter 5), and consider the following family of re-ranking criteria:

$$Cen(d; G) \cdot p_d(q), \tag{6.3}$$

where $d \in \mathcal{D}_{\text{init}}$, Cen is one of the centrality functions defined in the previous section, and $p_d(q)$ is the score that the initial retrieval engine assigns to d. This gives rise to the algorithms **Uniform Influx+LM** [6], **Weighted Influx+LM**, **Recursive Uniform Influx+LM**, and **Recursive Weighted Influx+LM**.

Incidentally, our choosing $p_d(q)$ as initial score function has the interesting consequence that it suggests interpreting $Cen(d; G)$ as a document "prior" — in fact, Lafferty and Zhai write, "with hypertext, [a document prior] might be the distribution calculated using the 'PageRank' scheme" (Lafferty and Zhai, 2001). We will return to this idea later.

6.3 Evaluation

In what follows, we first describe our language-model induction method. We then describe an array of experiments evaluating the effectiveness of our algorithms in re-ranking an initially retrieved list to improve precision at top ranks. We also explore via a controlled experimental setup the connection between centrality and relevance, and in doing so study the way relevant and non-relevant documents are situated within our graphs. We then compare our algorithms with previously proposed measures for inducing centrality that are based on document-specific properties. We analyze the importance of basing our graph formation on a language modeling framework and generation probabilities by studying an analogous framework using a vector-space representation and corresponding similarity measures. In addition, we examine another well-known Web-based algorithm for inducing centrality, HITS (Kleinberg, 1999), that in contrast to our algorithms is based on two different types of centrality (*hub* and *authority*) that are natural to the Web setting.

[6] Note that the ranking principle of the Uniform Influx+LM algorithm is the same as that of the bag-select algorithm (when implemented on top of our LM-based nearest-neighbor clusters) in Section 5.2 of Chapter 5.

6.3.1 Language model induction

For estimating $p_d(\cdot)$, treated as a similarity rather than a probability, we use the estimate $p_d^{KL,\mu}(\cdot)$ from Equation 2.5 in Section 2.3. As described in Section 2.3, for term sequence $\vec{t}$ the following equality holds:

$$p_d^{KL,\mu}(\vec{t}) = \exp\left(H(\vec{t})\right) \cdot p_d^{[\mu]}(\vec{t})^{\frac{1}{|\vec{t}|}}.$$

High entropy may be correlated with a larger number of unique terms — for example, we get an entropy of 0 for the document "Salvador Salvador Salvador" but $\log 3$ for "Toronto Sheffield Salvador" — which, in turn, has previously been suggested as a cue for relevance (Singhal et al., 1996; Hiemstra and Kraaij, 1999). Furthermore, in Section 6.3.5 we show that in the re-ranking setting, high entropy might be an indicator for relevance. Hence, generators of documents inducing high-entropy language models may be good candidates for centrality status. (We hasten to point out, though, that for the algorithms based on smoothed graphs (Definition 5), the entropy term cancels out due to our normalization of edge weights.)

6.3.2 Experimental setting

We use the same experimental setup as that of Section 5.3.2. Specifically, our choice of $\mathcal{D}_{\text{init}}$ ($|\mathcal{D}_{\text{init}}| = 50$) and its initial ranking, the optimized baselines and the smoothing parameter (μ) for the estimated language models are all described in Section 5.3.2 (page 61).

We choose the value of α (the "graph out-degree", that is, the ancestry parameter controlling the number of top generators considered for each document) from $\{2, 4, 9, 19, 29, 39, 49\}$; the search range for λ (the edge-weight smoothing factor) is $0.05, 0.1, 0.2, \ldots, 0.9, 0.95$. We optimize performance for prec@5; the optimization procedure for parameter tuning is the same as that described in Section 5.3.2.

In the results tables that follow, we use the following abbreviations for algorithm names.

U-In	Uniform Influx
W-In	Weighted Influx
R-U-In	Recursive Uniform Influx
R-W-In	Recursive Weighted Influx
U-In+LM	Uniform Influx+LM
W-In+LM	Weighted Influx+LM
R-U-In+LM	Recursive Uniform Influx+LM
R-W-In+LM	Recursive Weighted Influx+LM

Table 6.1: Primary experimental results, showing algorithm performance with respect to our 9 evaluation settings (3 performance metrics × 3 corpora). For each evaluation setting, improvements over the optimized baselines are given in italics; statistically significant differences between our structural re-ranking algorithms and the initial ranking and optimized baselines are indicated by *i* and *o* respectively; **bold** highlights the best results over all ten algorithms.

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
upper bound	.876	.788	.930	.944	.850	.980	.896	.800	1.000
init. ranking	.457	.432	.596	.500	.456	.691	.536	.484	.748
opt. baselines	.465	.439	.635	.512	.464	.696	.560	.494	.772
U-In	*.513*	*.492* $^{i}_{o}$	*.640*	.500	.442	.622	.512	.472	.673
W-In	*.515*	*.487* ^{i}	*.643*	.488	.432	.637	.520	.470	.644 $_{o}$
U-In+LM	*.509*	**.494** $^{i}_{o}$	.631	*.528*	**.518** $^{i}_{o}$	.665	.544	.490	.724
W-In+LM	*.511* ^{i}	*.486* $^{i}_{o}$	.630	*.516*	.464	**.703**	.560	**.500**	**.787**
R-U-In	*.513*	*.477*	.625	*.532*	.450	.687	.536	.478	.707
R-W-In	*.519*	*.480*	.632	*.524*	.446	.666	.536	.486	.699
R-U-In+LM	*.519* $^{i}_{o}$	*.491* $^{i}_{o}$	**.652**	*.556*	.460	.684	**.576** ^{i}	*.496*	.757
R-W-In+LM	**.531** $^{i}_{o}$	*.492* $^{i}_{o}$	.630	**.560**	.460	.676	*.572* ^{i}	*.496*	.747

6.3.3 Primary evaluations

Our main experimental results are presented in Table 6.1. The first three rows specify reference-comparison data. The initial ranking was, as described in Section 5.3.2, produced using $p_d^{KL,\mu}(q)$ with μ chosen to optimize for non-interpolated precision at 1000. We recall that the empirical upper bound on structural re-ranking, which applies to any algorithm that re-ranks $\mathcal{D}_{\text{init}}$, indicates the performance that would be achieved if all the relevant documents within the initial fifty were placed at the top of the retrieval list.

The first question we are interested in is how our graph-based algorithms taken as a whole do. As shown in Table 6.1, our methods improve upon the initial ranking in many cases, specifically, roughly 2/3 of the 72 relevant comparisons (8 centrality-based algorithms × 3 corpora × 3 evaluation metrics).

An even more gratifying observation is that Table 6.1 shows (via italics and boldface) that in many cases, our algorithms, even though optimized for precision at 5, can outperform a language model optimized for a different (albeit related) metric m even when performance is measured with respect to m; see, for example, the results for precision at 10 on the AP corpus.

Closer examination of the results in Table 6.1 reveals that in a majority of the evaluation settings for AP and TREC8, our algorithms are more effective when applied to the graph G_W than when applied to G_U. On WSJ, however, G_U seems to be a better choice. In general, the performance patterns imply that it is a bit better to explicitly incorporate generation probabilities into the edge weights of our generation graphs than to treat all the top generators of a document equally. (Experiments with the AP89 corpus in Kurland and Lee (2005) provide further support for this conclusion.)

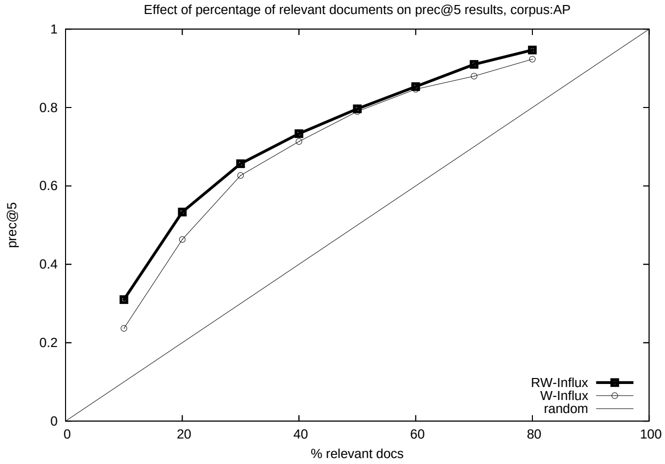
Another observation we can draw from Table 6.1 is that adding in query-generation probabilities as weights on the centrality scores (see Equation 6.3) tends to enhance performance. This can be seen by comparing rows labeled with some algorithm abbreviation “X” against the corresponding rows labeled “X+LM”: about 83% of the 36 relevant comparisons exhibit this improvement. Most of the counterexamples occur in settings involving precision at 10 and MRR, which we did not optimize our algorithms for.

Similarly, by comparing “Y”-labeled rows with “R-Y”-labeled ones, we see that in about 2/3 of the 36 relevant comparisons, it is better to use the recursive formulation of Equation 6.2, where the centrality of a document is affected by the centrality of its offspring, than to ignore offspring centrality as is done by Equation 6.1.

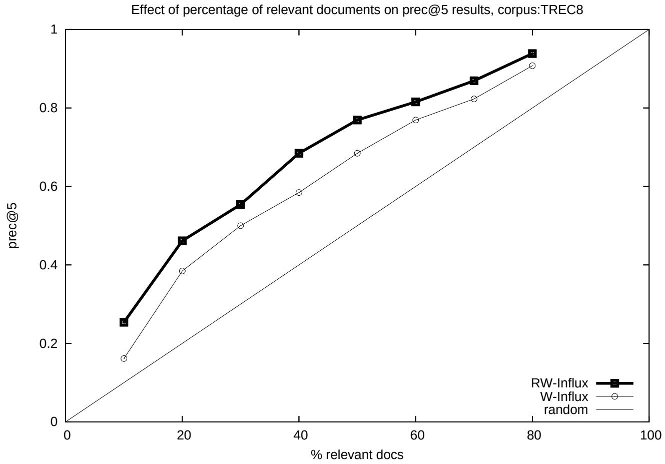
Perhaps not surprisingly, then, the Recursive Uniform Influx+LM and Recursive Weighted Influx+LM algorithms, which combine the two preferred features just described (recursive centrality computation and use of the initial search engine's score function) appear to be our best performing algorithms: working from a starting point below the optimized baselines, they improve the initial retrieval set to yield results that even at their worst, are not only clearly better than the initial ranking for precision at 5 and 10, but are also statistically indistinguishable from the optimized baselines. Moreover, on both AP and WSJ, the improvements they post over the initial ranking with respect to prec@5, the metric for which performance was optimized, are statistically significant. (In fact, on AP they post significant improvements over the optimized baselines for both prec@5 and prec@10).

Figure 6.2: Centrality and relevance: the effect of varying the percentage of relevant documents in $\mathcal{D}_{\text{init}}$ on the performance (prec@5) of the Weighted Influx and Recursive Weighted Influx algorithms.

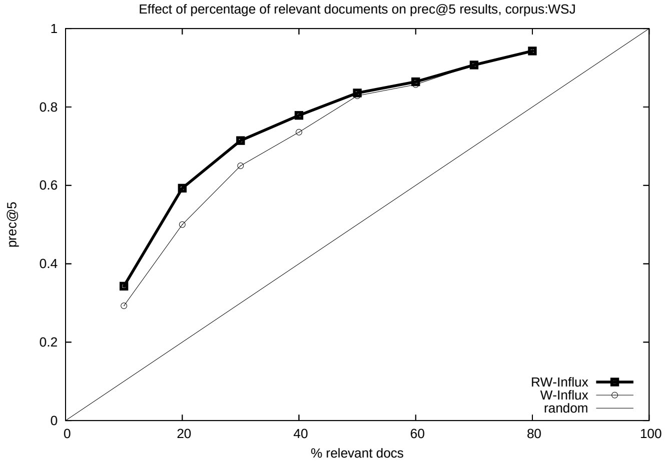
AP


Effect of percentage of relevant documents on prec@5 results, corpus:AP
prec@5
% relevant docs
0
0.2
0.4
0.6
0.8
1
0
20
40
60
80
100
RW-Influx
W-Influx
random

TREC8


Effect of percentage of relevant documents on prec@5 results, corpus:TREC8
prec@5
% relevant docs
0
0.2
0.4
0.6
0.8
1
0
20
40
60
80
100
RW-Influx
W-Influx
random

WSJ


Effect of percentage of relevant documents on prec@5 results, corpus:WSJ
prec@5
% relevant docs
0
0.2
0.4
0.6
0.8
1
0
20
40
60
80
100
RW-Influx
W-Influx
random

6.3.4 Centrality and relevance

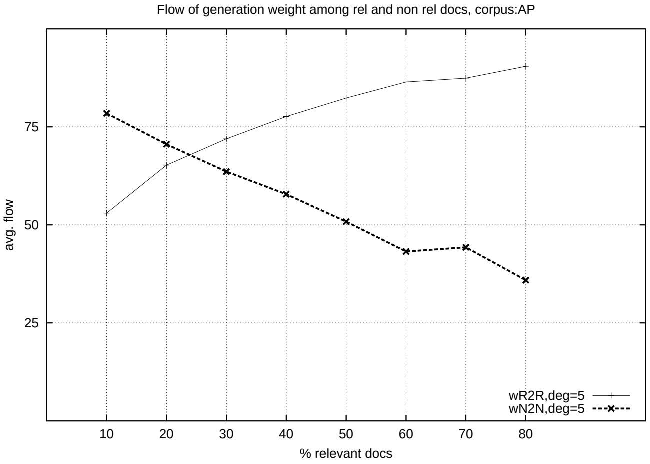
We now further explore the connection between centrality in $\mathcal{D}_{\text{init}}$ and relevance. In the previous section we saw that ranking solely based on centrality values (i.e., using the algorithms Uniform Influx, Weighted Influx, Recursive Uniform Influx and Recursive Weighted Influx) was effective to varying degrees. For example, while on the AP corpus the Uniform Influx and Weighted Influx algorithms post significant improvements (for prec@5 and prec@10) not only over the initial ranking but also over the optimized baselines, their performance is somewhat inferior to the initial ranking on WSJ and TREC8. One reason for such variation is the percentage of relevant documents in $\mathcal{D}_{\text{init}}$; this percentage exhibits high variance over different queries and corpora.

To factor out the differences in percentage of relevant documents in $\mathcal{D}_{\text{init}}$ across all queries and across all corpora, in order to evaluate the effect it has on the correlation between centrality and relevance, we establish the following experimental setting. For each query, we scan the original ranked list of documents (from which $\mathcal{D}_{\text{init}}$ was extracted) from the highest ranked document to the document at rank 1000, and collect n relevant documents. (Queries with fewer than n relevant documents among the 1000 are omitted; we experimented with $\text{n} = 5, 10, 20, 30, 40$.) Then, we perform another pass (top to bottom) and accumulate $50-n$ non-relevant documents. $\mathcal{D}_{\text{init}}$ is then the set of 50 collected documents, n of which are relevant.

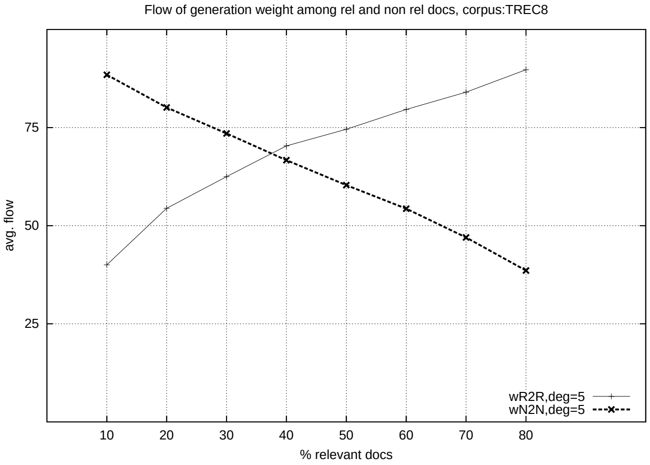
Note that our list construction is guided by the initial ranking in order to maintain the similarity patterns (with respect to the query) among the returned results as far as possible. We thus try to preserve (as much as possible) the concept of "re-ranking an initial list of documents retrieved in response to a query".

Figure 6.3: The percentage of generation weight on edges between a (non) relevant document and its 5 top generators that are (non) relevant (wR2R (wN2N)), with respect to the total generation weight on the document's outgoing edges (percentages are averaged over documents) as a function of the percentage of relevant documents in $\mathcal{D}_{\text{init}}$.

AP


Flow of generation weight among rel and non rel docs, corpus:AP
avg. flow
75
50
25
10
20
30
40
50
60
70
80
% relevant docs
wR2R,deg=5
wN2N,deg=5

TREC8


Flow of generation weight among rel and non rel docs, corpus:TREC8
avg. flow
75
50
25
10
20
30
40
50
60
70
80
% relevant docs
wR2R,deg=5
wN2N,deg=5

WSJ

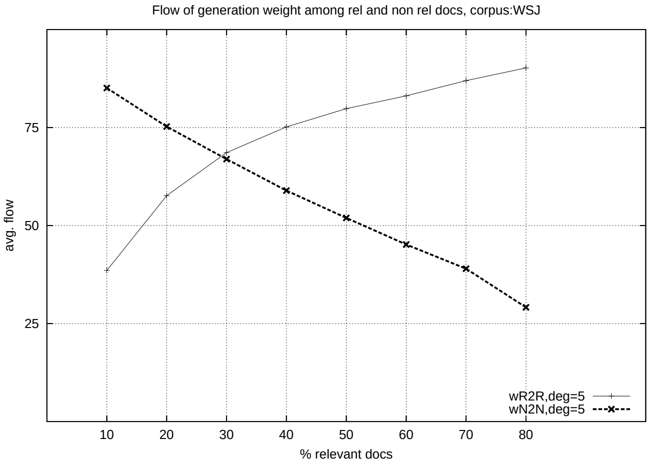

Flow of generation weight among rel and non rel docs, corpus:WSJ
avg. flow
75
50
25
10
20
30
40
50
60
70
80
% relevant docs
wR2R,deg=5
wN2N,deg=5

Figure 6.2 presents the prec@5 performance of our Weighted Influx and Recursive Weighted Influx algorithms when the percentage of relevant documents in $\mathcal{D}_{\text{init}}$ is varied as described above. (We present only prec@5 and the weighted-graphs-based algorithms so as to avoid cluttering the figures; results for prec@10 exhibit exactly the same patterns as those for prec@5.)

Our first observation with respect to Figure 6.2 is that for all three corpora, centrality as induced by either of the two presented algorithms is connected with relevance, as the performance curves of both algorithms are above the random line. We also note that for both algorithms (on all three corpora), the performance is monotonically increasing with respect to the percentage of relevant documents. (We hasten to point out that one would expect from any re-ranking method to exhibit performance that is monotonically non-decreasing with respect to the percentage of relevant documents in the initial list upon which re-ranking is performed.)

We also note that on TREC8, our algorithms performance is somewhat closer to random than on WSJ and AP. We attribute this to the fact that TREC8 is a much more heterogeneous corpus than the other two, and the queries which we tested for it are considered challenging (Hu et al., 2003).

In comparing the performance of the Weighted Influx and Recursive Weighted Influx algorithms in Figure 6.2, we clearly see that the latter is at least as effective as the former for all tested values of relevant document percentage in $\mathcal{D}_{\text{init}}$ and for all three corpora. This finding, again, supports the hypothesis that for determining the centrality of a document, the centrality of the documents from which it gets support is a very important source of information that should be considered. (See Equation 6.2 vs. Equation 6.1.)

Structure of $\mathcal{D}_{\text{init}}$ An interesting question, which naturally affects the correlation between centrality in $\mathcal{D}_{\text{init}}$ (as induced by our methods) and relevance, is the situation of relevant and non-relevant documents within the graphs we construct. More specifically, we would like to know whether the top generators of relevant documents tend to themselves be relevant, and whether the top generators of non-relevant documents tend to also be non-relevant. Such an analysis is closely related to van Rijsbergen's cluster hypothesis: *"Closely associated documents tend to be relevant to the same requests"* (van Rijsbergen, 1979, chapter 3). As mentioned before, this hypothesis has motivated many cluster-based retrieval approaches (e.g., Jardine and Rijsbergen (1971) and Croft (1980)) and has also been explored in the re-ranking setting (Leuski and Allan, 1998; Hearst and Pedersen, 1996; Tombros et al., 2002).

To perform this analysis, we measure the relative weights on edges from a (non-) relevant document to its top generators that are also (non-)relevant with respect to the total sum of weights on the document's outgoing edges. Indeed, this measure, when applied to the uniform-edge-weight graph G_U, is exactly Voorhees' (1985) cluster-hypothesis test applied to the re-ranking setting. Figure 6.3 presents the values of this estimate using the G_W graph (whose edge weights represent generation "probabilities") constructed with $\alpha = 5$ (i.e., we consider for each document its 5 top generators).[7] We present the resultant numbers as a function of the percentage of relevant documents in $\mathcal{D}_{\text{init}}$ (We control this percentage as described above.)

Our first observation from Figure 6.3 is that increasing the percent of relevant documents results in two effects. The flow of generation weight from relevant documents to relevant documents increases (as the "wR2R" line indicates), and therefore the flow of generation weight from relevant to non-relevant documents decreases. In addition, the flow of generation weight from non-relevant documents to relevant document increases (since the "wN2N" line goes down), and therefore flow from non-relevant to non-relevant documents decreases.

Perhaps the most important message rising from Figure 6.3 comes from observing the points in the graphs indicating that 50% of the documents in $\mathcal{D}_{\text{init}}$ are relevant. In these cases, more than 75% of the generation weight that a relevant document "transfers" to its top-generators goes to relevant documents, while about more than 45% of the generation weight that a non-relevant document "transfers" goes to relevant documents. Therefore, while the set of relevant documents in $\mathcal{D}_{\text{init}}$ "tends" to keep centrality support within the set, the set of non-relevant documents "leaks" a lot of centrality-status support to relevant documents. This observation helps to shed some light on our finding above that centrality (as induced by our methods) in $\mathcal{D}_{\text{init}}$ and relevance are correlated.

6.3.5 Non-structural re-ranking

So far, we have discussed the use of graph-based centrality as a re-ranking criterion, the idea being that relationships between documents can serve as an additional source of information. Our best empirical results in Table 6.1 seem to be produced by using the weighted formulation given in Equation 6.3 from Section 6.2.3:

$$Cen(d; G) \cdot p_d(q).$$

Since, as noted above, in this equation $Cen(d; G)$ can be regarded as a "prior" on documents, it is natural to ask whether other previously-proposed biases on generation probabilities might prove similarly useful. The comparison is especially interesting because these biases have tended to be isolated-document heuristics; we thus refer to their use as a replacement for $Cen(d; G)$ as "non-structural re-ranking".

Document length has been employed several times in the past to model the intuition that longer texts contain more information (Hiemstra and Kraaij, 1999; Kraaij and Westerveld, 2001; Miller et al., 1999). We refine this hypothesis to disentangle several distinct notions of information: the number of *tokens* in a

[7] Results for G_U exhibit the exact same patterns as those for G_W and are therefore omitted.

Table 6.2: Comparison between our use of language-model-based structural-centrality scores in Equation 6.3 vs. non-structural re-ranking heuristics. For each evaluation setting, *italics* mark improvements over the default baseline of uniform centrality scores, stars (*) indicate statistically significant differences with this default baseline, and **bold** highlights the best results over all eight algorithms.

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
uniform (= init)	.457	.432	.596	.500	.456	.691	.536	.484	.748
W-In	_.511*_	_.486*_	**.630**	_.516_	_.464_	_.703_	_.560_	**.500**	**.787**
R-W-In	__.531*__	__.492*__	_.630_	**.560**	_.460_	.676	__.572*__	_.496_	.747
length	.416	.414	.551	.472	.414	.642	.480	.446	.694
log(length)	.453	.432	_.606_	.496	_.468_	_.692_	_.552_	.484	.717
entropy	_.461_	.425	_.608_	.496	_.468_	__.717*__	_.544_	_.486_	.722
uniqTerms	.420	.413	.560	.492	.442	_.712_	.508	.456	.698
log(uniqTerms)	_.459_	.423	_.608_	.496	**.472**	_.700_	_.544_	_.490_	.723

document, the *distribution* of these tokens, and the number of *types* ("Salvador Salvador Salvador" contains three tokens but only one type). Thus, as substitutions for centrality in the above expression, we consider not only document length, but also the entropy of the term distribution and the number of unique terms, the latter statistic having served as the basis for pivoted unique normalization in Singhal et al. (1996). As baseline, we took the initial retrieval results; note that doing so corresponds to using a uniform bias, or, equivalently, using no bias at all.

As can be seen in Table 6.2, taking the log of token or type count is an improvement over using the raw frequencies, often yielding above-baseline performance. The entropy is more effective than raw frequency of either tokens or types, and in one case leads to the best performance overall. However, in the majority of settings, structural re-ranking gives the highest accuracies.

6.3.6 Information representation and similarity measures

We have advocated the use of generation relationships to define centrality, where these asymmetric relationships are based on language-model probabilities. To compare our choice with previously proposed notions of inter-document relationships, we first distinguish between two aspects. The first is *information representation*; in our framework, documents are represented via their induced unigram language models. Another well-known alternative is the *vector-space* representation (Salton et al., 1975). *Similarity measure* is the second aspect we have to consider. Models based on a vector-space representation often use the cosine as a (symmetric) similarity measure — indeed, as mentioned above, previous work in summarization (Erkan and Radev, 2004) used the cosine to determine centrality in ways very similar to the ones we have considered.

Figure 6.4 presents a comparison of the different methods we experimented with to define relevance-flow graphs. We focus on unigram language models and "log(tf).log(idf)"-weight vectors (Zobel and Moffat, 1998) for document representation[8], as they represent two approaches that have formed the basis for numerous approaches in information retrieval. While there is a huge number of similarity measures one can think of, we focus on a few simple but representative choices, rather than attempt to exhaustively search the enormous space of possible models of similarity.

Table (a) of Figure 6.4 provides the specification of the methods we compare, focusing on differences in choice of representation and similarity measure. The first two methods are based on a (smoothed) unigram language-model representation. The L method is the one we we have established our graph formulation on, i.e., language models with generation probabilities (details in Section 6.3.1). Method S utilizes the J divergence (Jeffreys, 1946), resulting in a symmetric variant of the generation probabilities in L: for two probability distributions p and q over terms, $J\left(p \mid\mid q\right) = D\left(p \mid\mid q\right) + D\left(q \mid\mid p\right)$.

The next three methods utilize the vector-space representation based on "log(tf).log(idf)" weights. In C, the cosine of the angle between the vector representation of d_1 and d_2 determines similarity, whereas in method T we use the inner product between the respective vectors. Method A presents an asymmetric variant of the previous two measures (we recall that $\cos(d_1, d_2) = \frac{d_1 \cdot d_2}{||d_1||_2 \cdot ||d_2||_2}$). It is interesting to note that this measure incorporates length normalization of d_2, which is similar in spirit to the "normalization" embedded in our estimates of language-model generation probabilities (refer back to Section 2.3 (Chapter 2) for details).

To run the evaluation for methods S, C, T and A we simply modify Definition 3 to use the corresponding similarity measure as the basis for determining edge weights of our graphs. Note that the fact that a measure is symmetric does not imply that edges (v_1, v_2) and (v_2, v_1) get the same weight even in our non-smoothed graphs — document d_1 being a top "generator" of d_2 with respect to the measure does not imply the reverse. It should also be observed that the language-model weights on centrality scores (i.e., the $p_d(q)$ term in Equation 6.3, on which the "+LM" algorithms are based) were *not* replaced with the similarity measure values, which makes sense since we want our comparison to focus on the effect of different means of computing graph-based centrality.

[8] Alternative representation utilizing tf.log(idf) weights resulted in inferior performance for the methods we examined.

Method	Information representation	Similarity measure (sim(d_1,d_2))	
L	language model	$\underline{\text{L}}$M generation probability	$\exp\left(-D(\widetilde{p}^{MLE}_{d_1}(\cdot)\vert\vert\widetilde{p}^{[\mu]}_{d_2}(\cdot))\right)$
S	language model	$\underline{\text{s}}$ymmetric LM-based measure	$\exp\left(-J\left(\widetilde{p}^{MLE}_{d_1}(\cdot)\ \Big\vert\Big\vert\ \widetilde{p}^{[\mu]}_{d_2}(\cdot)\right)\right)$
C	vector space	$\underline{\text{c}}$osine	$cos(d_1, d_2)$
T	vector space	inner produc$\underline{\text{t}}$	$d_1 \cdot d_2$
A	vector space	"$\underline{\text{a}}$symmetric cosine (inner product)"	$\frac{d_1 \cdot d_2}{\vert\vert d_2\vert\vert_2} = cos(d_1, d_2) \times \vert\vert d_1\vert\vert_2$

(a) Methods for forming relevance-flow graphs. Depending on the information representation approach, d_i ($i = 1, 2$) refers either to the term sequence that d_i is composed of (language model) or to d_i's vectorial representation (vector space). D and J denote KL divergence and J divergence respectively ($J\,(p \mid\mid q) = D\,(p \mid\mid q) + D\,(q \mid\mid p)$).

		U-In	W-In	U-In+LM	W-In+LM	R-U-In	R-W-In	R-U-In+LM	R-W-In+LM
AP	prec @5	$\boldsymbol{L}CAT$	$\boldsymbol{L}ACT$	$\hat{\boldsymbol{T}}LCA$	$\hat{\boldsymbol{L}}\hat{C}\hat{A}T$	$\hat{\boldsymbol{C}}\hat{A}LT$	$\boldsymbol{L}CAT$	$\hat{\boldsymbol{L}}\hat{C}\hat{A}\hat{T}$	$\hat{\boldsymbol{L}}\hat{C}\hat{A}T$
	prec @10	$\hat{\boldsymbol{L}}CTA$	$\hat{\boldsymbol{L}}CAT$	$\hat{\boldsymbol{L}}\hat{T}CA$	$\hat{\boldsymbol{L}}\hat{A}\hat{C}T$	$\hat{\boldsymbol{C}}\hat{A}LT$	$\hat{\boldsymbol{C}}\hat{A}LT$	$\hat{\boldsymbol{L}}\hat{C}\hat{A}\hat{T}$	$\hat{\boldsymbol{C}}\hat{A}\hat{L}\hat{T}$
	MRR	$\boldsymbol{L}$	$\boldsymbol{L}$	$\boldsymbol{L}$	$\boldsymbol{L}$		$\boldsymbol{L}$	$\boldsymbol{L}CA$	$\boldsymbol{L}CA$
TREC8	prec @5			$\boldsymbol{L}$		$\boldsymbol{L}$		$\boldsymbol{L}$	$\boldsymbol{L}CA$
	prec @10			$\hat{\boldsymbol{L}}CA$				$\hat{\boldsymbol{C}}\hat{A}$	
	MRR				$\boldsymbol{T}$				
WSJ	prec @5		$\boldsymbol{C}$	$\boldsymbol{C}\hat{T}A$	$\boldsymbol{A}CT$		$\boldsymbol{C}A$	$\hat{\boldsymbol{T}}\hat{C}\hat{A}\hat{L}$	$\boldsymbol{T}CA\hat{L}$
	prec @10				$\boldsymbol{C}A$				$\hat{\boldsymbol{C}}\hat{A}$
	MRR			$\hat{\boldsymbol{T}}$	$\boldsymbol{L}$			$\hat{\boldsymbol{C}}\hat{A}T$	

(b) Comparison of the different methods specified in Table (a) for relevance-flow-graph formation. In each entry, methods that pose a relative improvement of at least 5% over the initial ranking are ordered left to right in non-ascending order of performance (best performing method in each entry, i.e., the leftmost one, is boldfaced). A hat ("ˆ") marks methods that post a statistically significant improvement with respect to the initial ranking.

method	best	better than init. ranking	significantly better than init. ranking
L	**47.2%**	63.9%	16.7%
S	1.4%	26.4%	0%
C	36.1%	**72.2%**	**18.1%**
T	12.5%	56.9%	11.1%
A	2.8%	70.8%	**18.1%**

(c) Summary of the relative performance of methods in Table (a). For each method, the percentage of cases it performs the best, outperforms the initial ranking (by any margin) and significantly outperforms the initial ranking is presented. Percentages refer to the 72 relevant comparisons: 8 centrality-based algorithms × 3 corpora × 3 evaluation metrics. Bold: highest percent per column.

Figure 6.4: **Comparison of different methods for defining relevance-flow graphs.**

As can be seen in Table (c) of Figure 6.4, the L method (language-model generation probabilities) is in most cases the best performing method. Additional pairwise comparisons with the other methods revealed that indeed L is the best performing method. [9]

Table (c) shows that both the C (cosine) and A (its asymmetric variant) methods are very effective, as can be seen by the percentage of times they improve on the initial ranking (and significantly so). Pairwise comparisons between the two showed that neither of which managed to substantially outperform the other.

In comparing symmetric with asymmetric measures, we first see that our original proposal of asymmetric generation probabilities (L) is much more effective than its symmetric version (S) as can be seen in Tables (b) and (c). In comparing the inner product with its asymmetric variant (A), we see that the latter is more effective with respect to improvements over the initial ranking (and their significance). Furthermore, pairwise comparisons of the two methods give further support for the superiority of the asymmetric variant (A). However, as mentioned above, when comparing C and A — which is also C's asymmetric variant — with respect to pairwise comparisons, none of the two methods is superior to the other. (Although according to the "best performing" criterion one might suggest that C is superior.) We hasten to point, although, that our methods S and A are not necessarily the most effective ones for (a-)symmetrizing other measures.

Overall, while language-model generation probabilities indeed seem to be an attractive choice compared to other inter-document relationships considered in past literature, we believe that the important message emerging from our findings is that the overall graph-based structural re-ranking approach is a flexible and effective paradigm that can incorporate different types of inter-document relationships when appropriate.

6.3.7 Inducing centrality with the HITS algorithm

One well-known alternative method for computing centrality in a graph is the HITS algorithm (Kleinberg, 1998), originally proposed for Web search. There has been some work utilizing it for text summarization in non-Web domains as well (Mihalcea, 2004). The reason we have not yet discussed it in detail is that it differs conceptually from our proposed algorithms in an important way: two different

[9] On the AP89 corpus used in Kurland and Lee (2005), some of the vector-space-based methods (C and T) are in general superior to the L method. Results for AP89 are omitted from this chapter as many of the queries for AP89 have no relevant documents in $\mathcal{D}_{\text{init}}$. However, it is also important to note that our choice of weight function for the vector-space methods constitutes an "optimization" step. Case in point: using tf.log(idf) weights (instead of log(tf).log(idf) weights) results in vector-space-based performance substantially inferior to that of L in most cases. A possible analogous optimization step for L could be tuning the language-model smoothing parameter (see Section 2.3 of Chapter 2).

Table 6.3: Comparing the HITS-based algorithms. For each of the four algorithms, we evaluate using either authority (A) or hub-ness (H) as centrality score. An entry depicts those centrality scores, if any, that lead to performance superior to that of the initial ranking. The left-to-right ordering within an entry reflects descending performance; the more effective centrality score is **boldfaced**. A hat ("^") marks instances in which the improvement over the initial ranking is to a statistical significant degree.

		U-HITS	W-HITS	U-HITS+LM	W-HITS+LM
AP	prec @5	$\boldsymbol{A}$	$\boldsymbol{A}H$	$\hat{\boldsymbol{A}}H$	$\boldsymbol{A}H$
	prec @10	$\boldsymbol{A}H$	$\boldsymbol{A}H$	$\hat{\boldsymbol{A}}H$	$\boldsymbol{A}H$
	MRR	$\boldsymbol{A}$	$\boldsymbol{A}$	$\boldsymbol{A}H$	$\boldsymbol{A}$
TREC8	prec @5			$\boldsymbol{A}H$	$\boldsymbol{A}$
	prec @10			$\hat{\boldsymbol{A}}H$	$\boldsymbol{H}$
	MRR			$\boldsymbol{H}$	$\boldsymbol{H}$
WSJ	prec @5			$\boldsymbol{A}H$	$\boldsymbol{A}H$
	prec @10			$\boldsymbol{A}$	$\boldsymbol{A}$
	MRR			$\boldsymbol{H}$	$\boldsymbol{A}$

notions of centrality are identified, represented by *hub* and *authority* scores. While the concepts of hubs and authorities are highly suitable for Web-search scenarios, it is less clear whether it is useful in our setting to distinguish between the two.

Using our graph notation from section 6.2, we define the authority and hub score of a document to be:

$$\begin{aligned} Cen_{AUTH}(d,G) &\overset{def}{=} \textstyle\sum_{o \in \mathrm{Off}(d)} wt(o \rightarrow d) \cdot Cen_{HUB}(o,G); \\ Cen_{HUB}(d,G) &\overset{def}{=} \textstyle\sum_{g \in TopGen(d)} wt(d \rightarrow g) \cdot Cen_{AUTH}(g,G). \end{aligned}$$

Convergence of an iterative procedure to determine the scores is guaranteed under mild assumptions (Kleinberg, 1999) [10].

Similarly to the definitions of the (recursive) influx algorithms in section 6.2, we define the algorithms **U-HITS** (which sets $G = G_U$) and **W-HITS** (which sets $G = G_W$). In either case, we compute both $Cen_{AUTH}(d,G)$ and $Cen_{HUB}(d,G)$, but only one is chosen as the centrality score on which the actual ranking is based. The corresponding algorithms based on Equation 6.3 are termed **U-HITS+LM** and **W-HITS+LM** respectively; again, for each such algorithm, we may choose to use either hub or authority scores as centrality measures.

Performance comparison of the HITS-based algorithms are presented in Table 6.3. Each entry in the table depicts in descending order of performance which choices, if any, of centrality measure result in performance better than that of the initial ranking. The hats ("^") mark improvements that are statistically significant. By counting the number of appearances of A (authority) in Table 6.3,

[10] Strictly speaking, the algorithm and proof of convergence as originally presented (Kleinberg, 1998) need (trivial) modification to apply to edge-weighted graphs.

Table 6.4: Performance comparison of the HITS-based algorithms, utilizing **authority scores**, as implemented on unsmoothed and smoothed (S) graphs with the recursive influx algorithms. Underline: best performance per block (3 algorithms × evaluation measure). Boldface: best result per column.

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
U-HITS	.509	.482	.645	.500	.452	.668	.532	.474	.685
(S)U-HITS	.517	.480	.627	.536	.464	.636	.512	.474	.696
R-U-In	.513	.477	.625	.532	.450	**.687**	.536	.478	.707
W-HITS	.509	.486	.638	.440	.424	.648	.504	.464	.638
(S)W-HITS	.511	.471	.642	.488	.440	.637	.520	.470	.642
R-W-In	.519	.480	.632	.524	.446	.666	.536	.486	.699
U-HITS+LM	.519	.491	**.669**	.528	**.502**	.667	.544	.490	.702
(S)U-HITS+LM	.525	**.493**	.643	.544	.500	.683	.572	**.504**	.768
R-U-In+LM	.519	.491	.652	.556	.460	.684	**.576**	.496	.757
W-HITS+LM	.519	.492	.635	.504	.452	.674	.564	.490	.771
(S)W-HITS+LM	.527	.487	.629	.528	.462	.665	.568	.498	**.772**
R-W-In+LM	**.531**	.492	.630	**.560**	.460	.676	.572	.496	.747

we observe that authority scores lead to improvements over the initial ranking in about 55% of the relevant comparisons (4 algorithms × 3 evaluation measures × 3 corpora). Furthermore, if we focus on prec@5, the evaluation measure for which performance was optimized, authority scores lead to improvements in 8 out of 12 relevant comparisons. In addition, when examining the results for the U-HITS+LM and W-HITS+LM algorithms when authority scores act as a "prior" on the initial search engine's score ($p_d(q)$), we get 14 cases of improvement over the initial ranking out of 18 relevant comparisons. (We hasten to point out that while Recursive Uniform Influx and Recursive Weighted Influx scores correspond to a stationary distribution that could be loosely interpreted as a prior, in which case multiplicative combination with query likelihood is sensible (refer to Equation 6.3 of Section 6.2), it is not usual to assign a probabilistic interpretation to hub or authority scores. However, for completeness of the comparison to follow, we also experimented with HITS-based scores as "priors".) Thus, authority scores seems to be quite an effective centrality measure in our graph framework. In contrast, hub scores lead to improvements over the initial ranking in only 42% of the relevant comparisons, and in the vast majority of Table 6.3's entries H is positioned after A, indicating that the hub scores lead to performance inferior to that achieved by utilizing authority scores.

This finding strengthens our hypothesis in section 6.2.1 that effective structural re-ranking of $\mathcal{D}_{\text{init}}$ can be accomplished by identifying documents that are "good" generators of other documents; indeed, documents with high authority scores are "good" generators, while documents with high hub scores are offspring of such generators.

In further analysis, we compare the HITS-based algorithms, utilizing authority scores, to our recursive influx algorithms, since these were shown to be quite effective in Table 6.2 and since both paradigms employ recursion in their definitions of centrality. However, such a comparison might not be completely fair since the recursive influx algorithms are implemented on graphs wherein the edge weights were smoothed (refer to Definition 5 in Section 6.2), while the HITS-based ones are not. (The HITS algorithm does not require any smoothing of edge weights.) This difference might be quite important, however, since HITS assigns scores of zero to nodes that are not in the graph's largest connected component (with respect to positive-weight edges, considered to be bi-directional). Notice that an unsmoothed graph may have several connected components, whereas utilizing smoothing (as in Definition 5) ensures that each node has a positive-weight directed edge to every other node (self loops included). Additionally, the re-weighted version of HITS has provable stability properties (Ng et al., 2001).

We therefore define **(S)U-HITS**, **(S)W-HITS**, **(S)U-HITS+LM** and **(S) W-HITS+LM** to be the analogs of the algorithms defined above, when setting G to $G_U^{[\lambda]}$ and $G_W^{[\lambda]}$ instead of G_U and G_W respectively.

The performance results of the HITS-based algorithms (with both unsmoothed and smoothed graphs) compared to those of the recursive influx algorithms are presented in Table 6.4. Our first observation is that running the HITS-based algorithms on the smoothed graphs results (in most of the relevant comparisons) in better performance than that obtained by running them on the unsmoothed graphs. (Note that when comparing two consecutive "HITS" rows, the one with (S) contains larger numbers in most of the cases.) Another observation from Table 6.4 is that the recursive influx algorithms have better performance in general than that of the corresponding HITS-based algorithms whether run on unsmoothed or smoothed graphs, when judging by pairwise comparisons of the algorithms. (We point out, though, that while in comparison to the HITS-based algorithms implemented on smoothed graphs with weights representing generation probabilities the recursive influx algorithms are indeed superior (see the "W" algorithms), this is not the case for graphs having uniformly weighted edges (see the "U" algorithms)). Furthermore, the best performance in terms of prec@5 — the evaluation metric for which we optimized performance — is always obtained by one of the recursive influx algorithms.

All in all, our conclusion in light of the above observations is that although the HITS algorithm is based on concepts more suited to the Web environment than to our setting (non-hypertext documents), it can still provide an effective mechanism for inducing centrality (in the form of authority scores), attesting to the flexibility of our graph-based framework in accommodating different notions of centrality.

Table 6.5: Comparison of our algorithms' performance when implemented over graphs utilizing the estimate from Equation 6.4 (marked by the prefix [off]) — thereby based on the "seek for central offspring" approach — with that of their implementation on the original graphs from Section 6.2 (representing the "seek for central generators" approach.). Underline: best result per block. Boldface: best result per column.

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
U-In	51.3	49.2	64.0	50.0	44.2	62.2	51.2	47.2	67.3
[off]U-In	38.4	38.5	49.6	34.8	32.2	54.3	40.0	38.2	50.4
W-In	51.5	48.7	64.3	48.8	43.2	63.7	52.0	47.0	64.4
[off]W-In	38.8	37.2	51.9	34.0	33.4	53.7	40.0	39.2	49.2
U-In+LM	50.9	**49.4**	63.1	52.8	**51.8**	66.5	54.4	49.0	72.4
[off]U-In+LM	45.7	43.2	59.6	50.8	45.8	71.0	54.4	46.8	75.6
W-In+LM	51.1	48.6	63.0	51.6	46.4	70.3	56.0	**50.0**	**78.7**
[off]W-In+LM	43.6	42.8	55.5	50.4	44.4	68.5	53.6	46.0	69.1
R-U-In	51.3	47.7	62.5	53.2	45.0	68.7	53.6	47.8	70.7
[off]R-U-In	40.8	40.5	51.3	35.2	32.2	53.9	41.2	39.6	52.4
R-W-In	51.9	48.0	63.2	52.4	44.6	66.6	53.6	48.6	69.9
[off]R-W-In	41.0	40.3	53.0	34.4	33.4	52.7	40.4	38.6	50.6
R-U-In+LM	51.9	49.1	**65.2**	55.6	46.0	68.4	**57.6**	49.6	75.7
[off]R-U-In+LM	46.1	43.1	61.0	52.0	45.8	**71.1**	55.6	48.2	72.6
R-W-In+LM	**53.1**	49.2	63.0	**56.0**	46.0	67.6	57.2	49.6	74.7
[off]R-W-In+LM	46.3	44.2	59.5	50.4	46.2	69.3	56.0	48.0	68.3

6.3.8 Seeking generators versus seeking offspring

Throughout this chapter we advocated the approach of searching for central generators in the initial list $\mathcal{D}_{\text{init}}$, that is, seeking documents with language models highly probable to generate terms in many other (central) documents. Now, we explore the question of whether an alternative paradigm based on searching for central offspring documents — i.e, documents with term sequences that with high probability are generated by many documents (that are themselves central offspring) — can also result in effective algorithms for re-ranking $\mathcal{D}_{\text{init}}$.

In fact, we have already seen in the previous section some evidence for our original approach (searching for central generators) being more effective than looking for central offspring in the HITS context: authority scores served as a much better centrality measure than hub scores.

To examine the question of generators centrality vs. offspring centrality in the context of our two original approaches for centrality induction, i.e., (recursive) influx, we first define a new estimate given documents $d_1, d_2 \in \mathcal{D}_{\text{init}}$:

$$p_{d_1}^{off}(d_2) \overset{def}{=} p_{d_2}(d_1), \tag{6.4}$$

which we simply use in Definition 3 (page 75) to result in graphs describing "flow" from documents to their top offspring. (Note that the resultant graphs do not completely reflect an "edge inversion" of the original graphs, since the fact that g is a top generator of o does not imply that o is a top-offspring of g. (Recall our discussion in Section 6.2 with regard to the asymmetry embedded in our link induction method.) Observe also that both types of graphs have an α "out-degree" parameter.)

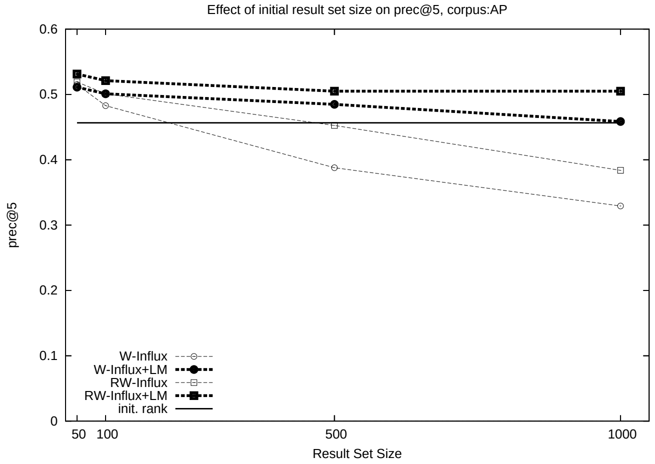
Thus, the Uniform Influx, Weighted Influx, Recursive Uniform Influx and Recursive Weighted Influx algorithms implemented over these graphs rank documents in $\mathcal{D}_{\text{init}}$ by their centrality as offspring. However, for the "+LM" algorithms (see Equation 6.3 in Section 6.2.3), we still use $p_d(q)$ as weight on the centrality value, as we are interested in only testing the effectiveness of the offspring-search paradigm as centrality-based induction approach.

Table 6.5 presents performance numbers of our algorithms when implemented on the newly constructed graphs — those describing "flow" from generators to offspring— (such algorithms are indicated with the [off] prefix), and their implementation on the original graphs — describing "flow" from offspring to "generators". (Note that the latter results are the ones from Table 6.1.)

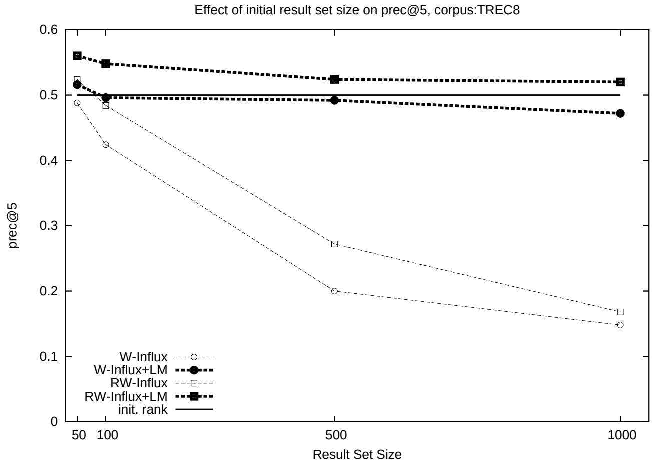
The message rising from Table 6.5 is clear: it is much better to use our original proposal of "searching for central generators" than to use the "searching for central offspring" approach, as for all algorithms, it is almost always the case that the implementation over the original graphs results in a substantially better performance than that obtained using the implementation over graphs constructed with the estimate from Equation 6.4. (Note that the underlined numbers almost always appear in rows with algorithms names not having the [off] prefix.).

Figure 6.5: The effect of varying the number of documents in $\mathcal{D}_{\text{init}}$ on the performance of the Weighted Influx, Recursive Weighted Influx, Weighted Influx+LM and Recursive Weighted Influx+LM algorithms performance (prec@5).

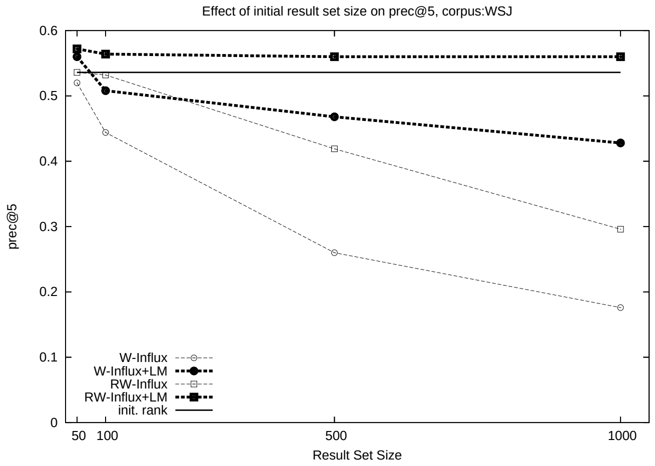
AP


Effect of initial result set size on prec@5, corpus:AP
prec@5
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
500
1000
Result Set Size
W-Influx
W-Influx+LM
RW-Influx
RW-Influx+LM
init. rank

TREC8


Effect of initial result set size on prec@5, corpus:TREC8
prec@5
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
500
1000
Result Set Size
W-Influx
W-Influx+LM
RW-Influx
RW-Influx+LM
init. rank

WSJ


Effect of initial result set size on prec@5, corpus:WSJ
prec@5
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
500
1000
Result Set Size
W-Influx
W-Influx+LM
RW-Influx
RW-Influx+LM
init. rank

6.3.9 The initial set size ($|\mathcal{D}_{\text{init}}|$)

We posed our centrality-computation techniques as methods for improving the results returned by an initial retrieval engine, and showed that they are successful at accomplishing this goal when the set of top retrieved documents ($\mathcal{D}_{\text{init}}$) is relatively small. (Recall that $|\mathcal{D}_{\text{init}}| = 50$). We now study the effect of including more documents from the initial ranking in $\mathcal{D}_{\text{init}}$ on the performance of our algorithms. Figure 6.5 presents the performance results of the Weighted Influx, Recursive Weighted Influx, Weighted Influx+LM and Recursive Weighted Influx+LM algorithms when the number of documents in $\mathcal{D}_{\text{init}}$ is varied. (We set $|\mathcal{D}_{\text{init}}|$ to values in $\{50, 100, 500, 1000\}$).

The first observation that can be made from Figure 6.5 is that the performance of the Weighted Influx and Recursive Weighted Influx algorithms, which use centrality values as the sole criterion for ranking, quickly degrades with increasing numbers of documents in $\mathcal{D}_{\text{init}}$. We do not see this finding as surprising, since such an increase necessarily results in a severe decrease in the ratio between the number of relevant documents to non-relevant documents in $\mathcal{D}_{\text{init}}$. As hypothesized in Section 6.2 and as was shown in Section 6.3.4, this ratio can have a major effect on the correlation between centrality and relevance.

We also observe in Figure 6.5 that the performance decrease of the Weighted Influx+LM and Recursive Weighted Influx+LM algorithms in light of an increase to the size of $\mathcal{D}_{\text{init}}$ is much more gradual than the one observed for the respective Weighted Influx and Recursive Weighted Influx algorithms. Furthermore, the Recursive Weighted Influx+LM algorithm posts performance that is better than that of the initial ranking for all tested values of $|\mathcal{D}_{\text{init}}|$ (on all three corpora). These performance patterns can be attributed to the initial ranking score embedded in the scoring functions of the Weighted Influx+LM and Recursive Weighted Influx+LM algorithms (recall Equation 6.3 from Section 6.2.3), which actually plays a dual role when increasing the size of $\mathcal{D}_{\text{init}}$: it lowers the final score of documents initially ranked low in the list — which therefore are less likely to be relevant — and helps to handle cases in which centrality and relevance are not strongly correlated (as explained in Section 6.2.3).

Another fact that is evident in Figure 6.5 is that the recursive algorithms post better performance than that of their non-recursive analogs for almost all values of $|\mathcal{D}_{\text{init}}|$. This gives further support to the observation that in computing the centrality of a document, the centrality of its offspring documents should be considered as well.

6.4 Related work

As described in Chapter 5, work on structural re-ranking in traditional ad hoc information retrieval has mainly focused on query-dependent clustering, wherein one seeks to compute and exploit a clustering of the initial retrieval results (Preece, 1973; Willett, 1985; Hearst and Pedersen, 1996; Leuski, 2001; Tombros et al., 2002;

Liu and Croft, 2004). Interestingly, some centrality measures have been previously employed to produce clusterings; in Tishby and Slonim (2000), the stochastic-process interpretation of our Equation 6.2 was utilized to detect structures in the underlying graph, thereby inducing a clustering.

In a related vein, Baliński and Daniłowicz (2005) and Diaz (2005) apply score regularization to ensure that similar documents within an initial retrieved list receive similar scores. In contrast to our framework, centrality is not introduced as an explicit notion and therefore cannot be explored at its own right.

There has been increasing use of techniques based on graphs induced by implicit relationships between documents or other linguistic items (Salton and Buckley, 1988; Hatzivassiloglou and McKeown, 1997; Daniłowicz and Baliński, 2000; Dhillon, 2001; Lafferty and Zhai, 2001; Joachims, 2003; Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Pang and Lee, 2004; Toutanova et al., 2004; Barzilay and Lapata, 2005; Collins-Thompson and Callan, 2005; Otterbacher et al., 2005; Zhang et al., 2005; Zhu, 2005; Erkan, 2006).

Work in the domain of text summarization (Erkan and Radev, 2004; Mihalcea and Tarau, 2004) resembles ours, in that it also computes centrality on graphs, although the nodes correspond to sentences or terms instead of documents. Specifically, Erkan and Radev (2004) and Mihalcea and Tarau (2004) present methods similar to our Recursive Weighted Influx algorithm, and the Uniform Influx algorithm is also used in (Erkan and Radev, 2004) for selecting central sentences. Perhaps the main contrast with our work is that links were not induced by generation probabilities, but by (symmetric) vector-space similarity measures; Section 6.3.6 presents the results of experiments studying the relative merits of our particular choice of link definition. Moreover, in Erkan and Radev (2004) and Mihalcea and Tarau (2004), summarization does not depend on a specific user's request and therefore centrality serves as the sole criterion for selecting sentences, while in the ad hoc retrieval setting, one has to handle cases in which relevance and centrality are not strongly correlated; one method for doing so is the technique represented by Equation 6.3.

Otterbacher et al. (2005) and Daniłowicz and Baliński (2000) use inter-item similarities to define transition probabilities of a Markov chain for sentence retrieval and document re-ranking respectively. While the the Markov chain approach is similar to our Recursive Weighted Influx method, the transition probabilities are based on symmetric inter-item similarity functions, in contrast to our link induction method. Furthermore, these probabilities also incorporate information about similarity to the information need at hand, and thus centrality cannot be explored in isolation as in our algorithms.

Recent work on Web retrieval (Zhang et al., 2005) utilizes asymmetric similarity relationships in the vector space for link induction. Centrality is induced using the PageRank algorithm, similarly to our Recursive Weighted Influx algorithm. However, since the task at hand is to rank all documents in the corpus (rather than re-rank an initially retrieved list), centrality is integrated with similarity to the query to define a relevance scoring function and is not explored in isolation.

In Section 6.3.6 we explored one alternative for utilizing an asymmetric similarity measure within the vector space model, studying its effectiveness both as an isolated ranking criterion and in combination with similarity to the query following Equation 6.3.

Our graphs are constructed using a nearest-neighbor principle. Such graphs have been used for numerous tasks (Levow and Matveeva, 2004; Matveeva, 2004; Zhu, 2005; Erkan, 2006). For document re-ranking (Levow and Matveeva, 2004; Matveeva, 2004), for example, a lower dimensional representation of the initial list is obtained using nearest-neighbor graphs, and queries are folded into the new space and then compared to the folded documents to obtain new scores used for re-ranking. Score regularization of documents in an initially retrieved list (Diaz, 2005) is another example of an application in which nearest-neighbor graphs are used. Some previous work (Belkin and Niyogi, 2003) showed that using Laplacian eigenmaps on such graphs can help to find a locality-preserving lower-dimensional manifold embedding of the original data.

Our centrality scores constitute a relationship-based re-ranking criterion that can serve as a bias affecting the initial retrieval engine's scores, as in Equation 6.3. Alternative biases that are based on individual documents alone have also been investigated. Functions incorporating document or average word length (Hiemstra and Kraaij, 1999; Kraaij and Westerveld, 2001; Miller et al., 1999) are applicable in our setting; we reported on experiments with (variants of) document length in Section 6.3.5. Other previously suggested biases that may be somewhat less appropriate for general domains include document source (Miller et al., 1999) and creation time (Li and Croft, 2003), as well as webpage hyperlink in-degree and URL form (Kraaij et al., 2002).

Chapter 7
Bipartite cluster-document relevance-flow graphs[1]

In this chapter we present algorithms for structural re-ranking that incorporate cluster information into the graph-based framework of Chapter 6. The main idea is to perform re-ranking based on centrality within bipartite graphs of documents (on one side) and clusters (on the other side), on the premise that these are mutually reinforcing entities. We find that cluster-document graphs give rise to much better retrieval performance than document-only graphs do.

For example, authority-based re-ranking of documents via a HITS-style cluster-based approach outperforms our previously-proposed Recursive Weighted Influx algorithm as applied to the solely-document graphs of Chapter 6. Moreover, we also show that computing authority scores for clusters constitutes an effective method for detecting clusters that contain a large percentage of relevant documents.

7.1 Re-consideration of HITS

In the previous chapter we saw that centrality in the initially retrieved list ($\mathcal{D}_{\text{init}}$) as induced by our Weighted Influx and Recursive Weighted Influx algorithms is highly correlated with relevance. (Refer back to Section 6.3.4 of Chapter 6.) Furthermore, the latter algorithm (Recursive Weighted Influx) — a weighted variant of PageRank (Brin and Page, 1998), which is based on recursive formulation of centrality— was shown to be more effective than the Weighted Influx algorithm, which is based on a weighted in-degree criterion.

The arguably most well-known alternative to PageRank is Kleinberg's *HITS* (hyper-text induced topic selection) algorithm (Kleinberg, 1998). As described in the previous chapter, the major conceptual way in which HITS differs from PageRank is that it defines two different types of central items: each node is assigned both a *hub* and an *authority* score as opposed to a single PageRank score. In the Web setting, in which HITS was originally proposed, good hubs correspond roughly to high-quality resource lists or collections of pointers, whereas good authorities correspond to the high-quality resources themselves; thus, distinguishing between two differing but interdependent types of Webpages is quite appropriate.

However, as described in Section 6.3.7 of Chapter 6, when applying the HITS-based algorithm to non-Web documents in our re-ranking setting, we found that its performance was was not as effective as that of the Recursive Weighted Influx-based algorithms. (This observation is true in comparing both ranking based solely on centrality, and ranking that incorporates the initial scores as per Equation 6.3.)

[1] This chapter is based on a paper written with Lillian Lee (Kurland and Lee, 2006) that will appear in the proceedings of SIGIR 2006.

Do these results imply that Recursive Weighted Influx (i.e., PageRank) is better than HITS for structural re-ranking of non-Web documents? Not necessarily, because there may exist graph-construction methods that are more suitable for HITS. Note that the only entities considered in the previous chapter were documents. If we could introduce entities distinct from documents but enjoying a mutually reinforcing relationship with them, then we might better satisfy the spirit of the hubs-versus-authorities distinction, and thus derive stronger results utilizing HITS.

A crucial insight of this chapter is that document *clusters* appear extremely well-suited to play this complementary role. The intuition is that: (a) given those clusters that are "most representative" of the user's information need, the documents within those clusters are likely to be relevant; and (b) the "most representative" clusters should be those that contain many relevant documents. This apparently circular reasoning is strongly reminiscent of the inter-related hubs and authorities concepts underlying HITS. Furthermore, we saw in Chapter 5 that clusters are a promising source of information in the re-ranking setting.

In this chapter, we show through an array of experiments that consideration of the mutual reinforcement of clusters and documents in determining centrality can lead to highly effective algorithms for re-ranking the initially retrieved list $\mathcal{D}_{\text{init}}$. Specifically, our experimental results show that the centrality-induction methods that we studied solely in the context of document-only graphs in Chapter 6 result in much better re-ranking performance if implemented over bipartite graphs of documents (on one side) and clusters (on the other side). For example, ranking *documents* by their "authoritativeness" as computed by HITS upon these cluster-document graphs yields better performance than that of our Recursive Weighted Influx algorithm applied to document-only graphs. Interestingly, we also find that *cluster* authority scores can be used to identify clusters containing a large percentage of relevant documents.

7.2 Re-ranking algorithms

As in Chapters 5 and 6, we are focused on the structural re-ranking paradigm, and therefore our algorithms are applied not to the entire corpus, but to the set $\mathcal{D}_{\text{init}}$ that was retrieved in response to the query q, by a given initial retrieval engine. Some of our algorithms also take into account the set $Cl(\mathcal{D}_{\text{init}})$ of *clusters* of the documents in $\mathcal{D}_{\text{init}}$, as was the case in Chapter 5. We use $\mathcal{S}_{\text{init}}$ to refer generically to whichever set of entities — either $\mathcal{D}_{\text{init}}$ or $\mathcal{D}_{\text{init}} \cup Cl(\mathcal{D}_{\text{init}})$ — is used by a given algorithm.

Following Section 6.2, we consider relevance-flow graphs that can be all represented as weighted directed graphs of the form (V, wt), where V is a finite non-empty set of nodes and $wt : V \times V \to \{y \in \Re : y \geq 0\}$ is a non-negative edge-weight function. Recall that while such graphs technically have edges between all ordered pairs of nodes (self-loops included), edges with zero edge-weight are conceptually equivalent to missing edges.

In what follows, we mainly focus on algorithms that rank documents in $\mathcal{D}_{\text{init}}$ based solely on centrality. Section 7.4 presents results for integrating centrality scores with the initial search engine scores as in Equation 6.3 of Chapter 6.

HITS The HITS algorithm (previously discussed in Section 6.3.7) for computing centrality can be motivated as follows. Let $G = (V, wt)$ be the input graph, and let v_2 be a node in V. First, suppose we somehow knew the *hub score* $Cen_{HUB}(v_1, G)$ of each node $v_1 \in V$, where "hubness" is the extent to which the nodes that v_1 points to are "good" in some sense. Then, v_2's *authority score*

$$Cen_{AUTH}(v_2, G) = \sum_{v_1 \in V} wt(v_1 \rightarrow v_2) \cdot Cen_{HUB}(v_1, G) \tag{7.1}$$

would be a natural measure of how "good" v is, since a node that is "strongly" pointed to by high-quality hubs (which, by definition, tend to point to "good" nodes) receives a high score. But where do we get the hub score for a given node v_1? A natural choice is to use the extent to which v_1 "strongly" points to highly authoritative nodes:

$$Cen_{HUB}(v_1, G) = \sum_{v_2 \in V} wt(v_1 \rightarrow v_2) \cdot Cen_{AUTH}(v_2, G). \tag{7.2}$$

Clearly, Equations 7.1 and 7.2 are mutually recursive , but the iterative HITS algorithm provably converges to (non-identically-zero, non-negative) score functions $Cen^*_{HUB}(\cdot, G)$ and $Cen^*_{AUTH}(\cdot, G)$ that satisfy the above pair of equations. (As mentioned in Section 6.3.7, the algorithm and proof of convergence as originally presented (Kleinberg, 1998) need (trivial) modification to apply to edge-weighted graphs.)

Figure 7.1 depicts the "iconic" case in which the input graph G is *one-way bipartite*, that is, V can be partitioned into non-empty sets V_{Left} and V_{Right} such that only edges in $V_{\text{Left}} \times V_{\text{Right}}$ can receive positive weight, and $\forall v_1 \in V_{\text{Left}}$, $\sum_{v_2 \in V_{\text{Right}}} wt(v_1 \rightarrow v_2) > 0$.

It is the case that $Cen^*_{AUTH}(v_1, G) = 0$ for every $v_1 \in V_{\text{Left}}$ and $Cen^*_{AUTH}(v_2, G) = 0$ for every $v_2 \in V_{\text{Right}}$; in this sense, the left-hand nodes are "pure" hubs and the right-hand nodes are "pure" authorities.

Recall that in the end, we need to produce a *single* centrality score for each node $n \in V$. As in Section 6.3.7 we consider only two possibilities in this chapter — using $Cen^*_{AUTH}(\cdot, G)$ as the final centrality score, or using $Cen^*_{HUB}(\cdot, G)$ instead— although combining the hub and authority scores is also an interesting possibility.

7.2.1 Graph schemata: incorporating clusters

Recall that the fundamental operation in our structural re-ranking paradigm is to compute the centrality of entities (with)in a set $\mathcal{S}_{\text{init}}$. One possibility is to define

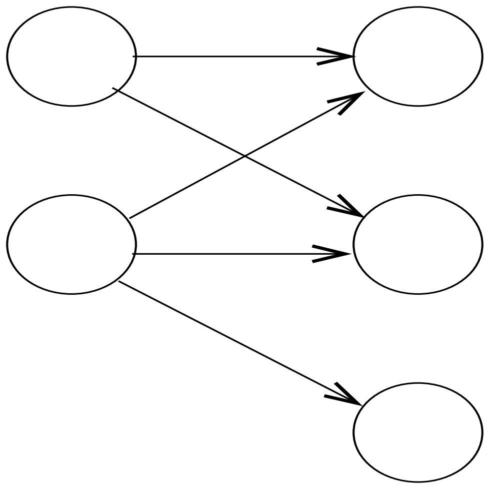


Figure 7.1: A one-way bipartite graph. We only show positive-weight edges (omitting weight values). According to HITS, the left-hand nodes are (pure) hubs; the right-hand ones are (pure) authorities.

$\mathcal{S}_{\text{init}}$ as $\mathcal{D}_{\text{init}}$, the documents in the initially retrieved set, as we did in Chapter 6; we refer to any relevance-flow graph induced under this choice as a ***document-to-document*** graph. But as previously discussed, for non-Web documents it may not be obvious *a priori* what kinds of documents are hubs and what kinds are authorities.

Alternatively, we can define $\mathcal{S}_{\text{init}}$ as $\mathcal{D}_{\text{init}} \cup Cl(\mathcal{D}_{\text{init}})$, where $Cl(\mathcal{D}_{\text{init}})$ consists of clusters of the documents in $\mathcal{D}_{\text{init}}$. On a purely formal level, doing so allows us to map the hubs/authorities duality discussed above onto the documents/clusters duality, as follows. Recalling our discussion of the "iconic" case of one-way bipartite graphs $G = ((V_{\text{Left}}, V_{\text{Right}}), wt)$, we can create ***document-as-authority*** graphs simply by choosing $V_{\text{Left}} = Cl(\mathcal{D}_{\text{init}})$ and $V_{\text{Right}} = \mathcal{D}_{\text{init}}$, so that necessarily clusters serve the role of (pure) hubs and documents serve the role of (pure) authorities. Contrariwise,[2] we can create ***document-as-hub*** graphs by setting $V_{\text{Left}} = \mathcal{D}_{\text{init}}$ and $V_{\text{Right}} = Cl(\mathcal{D}_{\text{init}})$.

But the advantages of incorporating cluster-based information are not just formal. The well-known *cluster hypothesis* (van Rijsbergen, 1979) encapsulates the intuition that clusters can reveal groups of relevant documents; in practice, the potential utility of clustering for this purpose has been demonstrated a number of times in the re-ranking setting (see Chapter 5). Since central clusters are, supposedly, those that accrue the most evidence for relevance, documents that are strongly identified with such clusters should themselves be judged highly relevant.[3]

[2] In practice, one can simultaneously compute the output of HITS for a given document-as-authority and document-as-hub graph *pair* by "overlaying" the two into a single graph and suitably modifying HITS's normalization scheme.

[3] We say "are strongly identified with", as opposed to "belong to" to allow

[4] But identifying such clusters is facilitated by knowledge of which documents are most likely to be relevant — exactly the mutual reinforcement property that HITS was designed to leverage.

7.2.2 Scoring by (recursive) influx

We will compare the results of using the HITS algorithm against those derived using the Recursive Influx scoring method instead (see Equation 6.2, page 77), as we saw in Chapter 6 that the latter resulted in effective algorithms (Recursive Uniform Influx, Recursive Weighted Influx) for document-only graphs [5].

Note that one can think of the Recursive Influx method (Equation 6.2):

$$Cen_{RI}(v_2; G) = \sum_{v_1 \in V} wt(v_1 \rightarrow v_2) \cdot Cen_{RI}(v_1; G),$$

as a version of HITS in which the hub/authority distinction has been collapsed. Thus, we can *conceptually* get the above equation by writing $Cen_{RI}(\cdot; G)$ for both $Cen_{HUB}(\cdot, G)$ and $Cen_{AUTH}(\cdot, G)$,

Recall that in Section 6.2, we used the PageRank smoothing scheme (Brin and Page, 1998) (see Definition 5) to alter the original graph $G = (V, wt)$ weight function (wt), thus taking care that the resultant graph $G^{[\lambda]} = (V, wt^{[\lambda]})$ represents an ergodic Markov chain for which the Recursive Influx equation from above has a solution. Alternatively, we can use the original graph $G = (V, wt)$ and simply alter the scoring function to ensure that the sum of weights on outgoing edges from each node is 1. One also has to apply a correction for nodes with no positive-weight edges emanating from them (Ng et al., 2001; Langville and Meyer, 2005) — which are abundant in one-way bipartite graphs since under the original "random surfer" model, the sum of the transition probabilities out of such nodes would be $(1 - \lambda)$, not 1. We therefore re-define our Recursive Influx ranking method over the "unsmoothed" graph G as:

for overlapping or probabilistic clusters. Indeed, the one-way bipartite graphs we construct are ill-suited to the HITS algorithm if document-to-cluster links are based on membership in disjoint clusters.

[4] This is, in some sense, a type of smoothing: a document might be missing some of the query terms (perhaps due to synonymy), but if it lies within a sector of "document space" containing many relevant documents, it could still be deemed highly relevant. The aspect-f and interpolation-f algorithms from Chapter 5, for example, are based on this smoothing idea. See also Liu and Croft (2004) for cluster-based smoothing in the re-ranking setting.

[5] Recall that Recursive Uniform Influx is essentially the PageRank algorithm (Brin and Page, 1998) and that Recursive Weighted Influx is a weighted analog of PageRank.

$$\begin{aligned} Cen^{[\lambda]}{}_{RI}(v_2;G) & \overset{def}{=} & \sum_{v_1 \in V: out(v_1)>0} \left[\frac{(1-\lambda)}{|V|} + \lambda \frac{wt(v_1 \to v_2)}{out(v_1)}\right] \cdot Cen^{[\lambda]}{}_{RI}(v_1;G) \\ & + & \sum_{v_1 \in V: out(v_1)=0} \frac{1}{|V|} \cdot Cen^{[\lambda]}{}_{RI}(v_1;G) \end{aligned} \quad (7.3)$$

where $out(v_1) \overset{def}{=} \sum_{v_2' \in V} wt(v_1 \to v_2')$, and $\lambda \in (0,1)$ is a parameter. (Conceptually, the role of the second summation in Equation 7.3 is to set $\lambda = 0$ for no-outflow nodes.)

Equation 7.3 is recursive, but there are iterative algorithms that provably converge to the unique positive solution $Cen^{*[\lambda]}_{RI}(\cdot;G)$ satisfying the sum-normalization constraint $\sum_{v \in V} Cen_{RI}(v;G) = 1$ (Langville and Meyer, 2005).

Moreover, note that setting $G = G_W$ (refer back to Section 6.2 for the definition of G_W) in Equation 7.3 and observing that G_W has no vertices with zero "outflux", we get that the solution to Equation 7.3 ($Cen^{*[\lambda]}_{RI}(\cdot;G)$) is exactly the scoring function used by our Recursive Weighted Influx algorithm from Chapter 6.

Moreover, a (non-trivial) *closed-form* — and quite easily computed — solution exists for one-way bipartite graphs:

Theorem 1. *If $G = (V, wt)$ is one-way bipartite, then*

$$Cen_{RI,Bip}(v_2;G) \overset{def}{=} \sum_{v_1 \in V: out(v_1)>0} \frac{wt(v_1 \to v_2)}{out(v_1)} \quad (7.4)$$

is an affine transformation (with respect to positive constants) of, and therefore equivalent for ranking purposes to, the unique positive sum-normalized solution to Equation 7.3.

Proof. Let the underlying one-way bipartite graph be $G = ((V_{\text{Left}}, V_{\text{Right}}), wt)$ and assume that $|V_{\text{Left}}| > 0$ and $|V_{\text{Right}}| > 0$. Let $N = |V_{\text{Left}} \cup V_{\text{Right}}|$. Let v_r denote a node in V_{Right} and v_l denote a node in V_{Left}.

For the one-way bipartite graph G, Equation 7.3 can be rewritten for any node $z \in V_{\text{Left}} \cup V_{\text{Right}}$ as

$$\begin{aligned} Cen_{RI}(z) & = & \sum_{v_l \in V_{\text{Left}}} \left[\frac{(1-\lambda)}{|V|} + \lambda \frac{wt(v_l \to z)}{out(v_l)}\right] \cdot Cen_{RI}(v_l) \\ & + & \sum_{v_r \in V_{\text{Right}}} \frac{1}{|V|} \cdot Cen_{RI}(v_r) \end{aligned} \quad (7.5)$$

(For clarity, we use the shortened notation $Cen_{RI}(z)$ to denote $Cen^{[\lambda]}{}_{RI}(z;G)$ throughout the proof.)

A unique positive and sum-normalized solution $Cen^{*}_{RI}(\cdot)$ to this set of equations exists because it describes the stationary distribution of an ergodic Markov chain.

Therefore, $S_R \overset{def}{=} \sum_{v_r \in V_{\text{Right}}} Cen^*_{RI}(v_r)$ exists and is positive, and $\sum_{v_l \in V_{\text{Left}}} Cen^*_{RI}(v_l) = 1 - S_R$.

Computing $Cen^*_{RI}(v_l)$ for $v_l \in V_{\text{Left}}$ from the equation above yields, after some algebra,

$$Cen^*_{RI}(v_l) = \frac{1 - \lambda + \lambda S_R}{N} + \sum_{v'_l \ \in V_{\text{Left}}} 0.$$

Denote $\kappa = \frac{1-\lambda+\lambda S_R}{N}$ $(\kappa > 0)$, we get

$$Cen^*_{RI}(v_l) = \kappa + \kappa\lambda \sum_{v'_l \in V_{\text{Left}}} \frac{wt(v'_l \rightarrow v_l)}{out(v'_l)},$$

since the second term is 0.

Using the value $Cen^*_{RI}(v_l)$ in Equation 7.5, we find using some algebra that

$$Cen^*_{RI}(v_r) = \kappa + \kappa\lambda \sum_{v'_l \in V_{\text{Left}}} \frac{wt(v'_l \rightarrow v_r)}{out(v'_l)}.$$

□

Interestingly, this result shows that while one might have thought that clusters and documents would "compete" for centrality score — as assigned by the Recursive Influx method — when placed within the same graph, in our document-as-authority and document-as-hub graphs this is not the case.

In addition, we will also examine the performance of our Influx method, which scores node v_2 by

$$Cen_I(v_2; G) = \sum_{v_1 \in V} wt(v_1 \rightarrow v_2).$$

This can be viewed as a non-recursive version of the Recursive Influx method (refer back to the discussion in Chapter 6) , or as an un-normalized analog of Equation 7.4.

7.2.3 Algorithms based on centrality scores

Clearly, we can rank documents by their scores as computed by any of the functions introduced above. But when we operate on document-as-authority or document-as-hub graphs, centrality scores for the clusters are also produced. These can be used to derive alternative means for ranking documents. We follow Liu and Croft's approach (Liu and Croft, 2004): first, rank the documents within (or most strongly associated to) each cluster according to the initial retrieval engine's scores; then, derive the final list by concatenating the within-cluster lists in order of decreasing cluster score, discarding repeats. Such an approach would be successful if cluster centrality is strongly correlated with the property of containing a large percentage of relevant documents.

Ranking algorithms Since we have two possible ranking paradigms, and our algorithms operate either on document-only graphs or on cluster-doc graphs we adopt the following **algorithm naming abbreviations** for this chapter. Abbreviations consist of a hyphen-separated prefix and suffix. The prefix ("doc" (or "d") or "clust" (or "c")) indicates whether documents were ranked directly by their centrality scores, or indirectly through the concatenation process outlined above in which it is the *clusters'* centrality scores that were employed. The suffix ("WAuth", "WHub", "RWI", or "WI") indicates which score function ($Cen^*_{AUTH}(\cdot, G)$, $Cen^*_{HUB}(\cdot, G)$, $Cen^{*[\lambda]}_{RI}(\cdot; G)$ (or $Cen_{RI,Bip}(\cdot; G)$), or $Cen_I(\cdot; G)$) was used to measure centrality. [6] For a given re-ranking algorithm, we indicate the graph upon which it was run in brackets, e.g., "doc-Auth[G]".

7.3 Evaluation framework

7.3.1 Graph construction

Relevance flow based on language models (LMs) To evaluate "relevance flow" between two nodes in our graphs (i.e., to determine edge weights) we follow Section 6.3.1 and utilize the language-model-based asymmetric similarity measure $p^{KL,\mu}_x(\cdot)$ for both documents and clusters.[7]

We extend the definition of top-generators ($TopGen(\cdot)$) from Definition 3 for settings with multiple types of entities. Specifically, we define $TopGen(x \mid m, R)$, to be the m items y within the "restriction set" R that have the highest values of $p^{KL,\mu}_y(x)$ (we break ties by item ID, assuming that these have been assigned to documents and clusters).

Graphs used in experiments For a given set $\mathcal{D}_{\text{init}}$ of initially retrieved documents and positive integer α (an "out-degree" parameter), we consider the following three graphs. Each connects nodes v_1 to the α other nodes, drawn from some specified set, that v_1 has the highest relevance flow to.

The document-to-document **d↔d** graph — denoted in the previous Chapter

[6] We use "W" in the suffixes since all our algorithms (in this chapter) are implemented over graphs wherein edge weights represent generation probabilities. (See Section 7.3).

[7] Recall from sections 2.3 and 5.3, that we treat a cluster as the big document resulting from the concatenation of its constituent documets. Concatenation order is irrelevant for unigram LMs.

as G_W [8] — has vertex set $\mathcal{D}_{\text{init}}$ and weight function

$$wt^{\text{d}\leftrightarrow\text{d}}(v_1 \to v_2) = \begin{cases} p_{v_2}^{KL,\mu}(v_1) & \text{if } v_2 \in \mathit{TopGen}(v_1 \,|\, \alpha, \mathcal{D}_{\text{init}} - \{v_1\}), \\ 0 & \text{otherwise.} \end{cases}$$

The document-as-authority graph **c→d** has vertex set $\mathcal{D}_{\text{init}} \cup \mathit{Cl}(\mathcal{D}_{\text{init}})$ and a weight function such that positive-weight edges go only from clusters to documents:

$$wt^{\text{c}\rightarrow\text{d}}(v_1 \to v_2) = \begin{cases} p_{v_2}^{KL,\mu}(v_1) & \text{if } v_1 \in \mathit{Cl}(\mathcal{D}_{\text{init}}) \text{ and} \\ & \quad v_2 \in \mathit{TopGen}(v_1 \,|\, \alpha, \mathcal{D}_{\text{init}}), \\ 0 & \text{otherwise.} \end{cases}$$

The document-as-hub graph **d→c** has vertex set $\mathcal{D}_{\text{init}} \cup \mathit{Cl}(\mathcal{D}_{\text{init}})$ and a weight function such that positive-weight edges go only from documents to clusters:

$$wt^{\text{d}\rightarrow\text{c}}(v_1 \to v_2) = \begin{cases} p_{v_2}^{KL,\mu}(v_1) & \text{if } v_1 \in \mathcal{D}_{\text{init}} \text{ and} \\ & \quad v_2 \in \mathit{TopGen}(v_1 \,|\, \alpha, \mathit{Cl}(\mathcal{D}_{\text{init}})), \\ 0 & \text{otherwise.} \end{cases}$$

Since the latter two graphs are one-way bipartite, Theorem 1 applies to them.

Clustering Method We utilize our language-model-based nearest-neighbor clustering method from Section 5.2 and define for each document d the cluster $\{d\} \cup \mathit{TopGen}(d \,|\, k-1, \mathcal{D}_{\text{init}} - \{d\})$, where k is the cluster-size parameter.

7.3.2 Experimental setup

We follow the exact same experimental set up as in Sections 5.3 and 6.3.2 for (a) setting $\mathcal{D}_{\text{init}}$, (b) choosing the optimized baselines, (c) setting the language-model smoothing parameter value and (d) optimizing different parameters . Thus, the values chosen for the free parameters in our algorithms are: (i) the graph "out-degree" α: $\{2, 4, 9, 19, 29, 39, 49\}$, (ii) the cluster size k: $\{2, 5, 10, 20, 30\}$, and (iii) the parameter λ (for the Recursive Influx-based algorithms): $\{0.05, 0.1 \ldots 0.9, 0.95\}$.

We also recall that a ranking method might assign different items the same score; we break such ties by item ID. Alternatively, the scores used to determine $\mathcal{D}_{\text{init}}$ can be utilized, if available.

[8] We use here the notation d↔d instead of G_W (used in Chapter 6) to differentiate this document-only graph from the bipartite graphs introduced in this chapter.

7.4 Experimental results

As in previous chapters, when we say that results or the difference between results are "significant", we mean according to the two-sided Wilcoxon test at a confidence level of 95%.

Recall that an algorithm is defined by the underlying method and the graph upon which it is implemented. Table 7.1 presents the algorithms that we discuss in what follows.

Table 7.1: Graph-based algorithms for structural re-ranking.

algorithm	method	graph
doc-WI[d↔d]	Weighted Influx	d↔d
doc-RWI[d↔d]	Recursive Weighted Influx	d↔d
doc-WAuth[d↔d]	W-HITS (using **authority** scores)	d↔d
doc-WAuth[d↔d]	W-HITS (using **hub** scores)	d↔d
doc-WI[c→d]	Weighted Influx	c→d
doc-RWI[c→d]	Recursive Weighted Influx	c→d
doc-WAuth[c→d]	W-HITS (using **authority** scores)	c→d
clust-WHub[c→d]	W-HITS (using **hub** scores)	c→d
clust-WI[d→c]	Weighted Influx	d→c
clust-RWI[d→c]	Recursive Weighted Influx	d→c
clust-WAuth[d→c]	W-HITS (using **authority** scores)	d→c
doc-WHub[d→c]	W-HITS (using **hub** scores)	d→c

As mentioned above, in some cases we will use the prefixes "d" and "c" instead of "doc" and "clust" respectively.

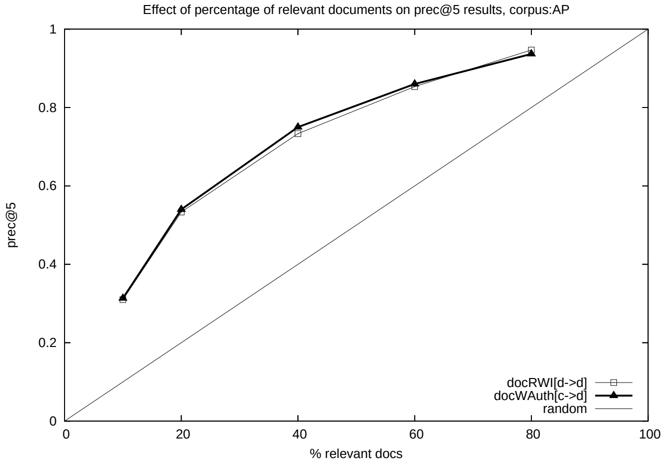
7.4.1 Re-ranking by document centrality

Main result We first consider our main question: can we substantially boost the effectiveness of HITS by applying it to cluster-to-document graphs, which we have argued are more suitable for it than the document-to-document graphs we constructed in the previous chapter? The answer, as shown in Table 7.2, is clearly "yes": we see that *moving to cluster-to-document graphs results in substantial improvement for HITS, and indeed boosts its results over those for Recursive Weighted Influx on document-to-document graphs.*

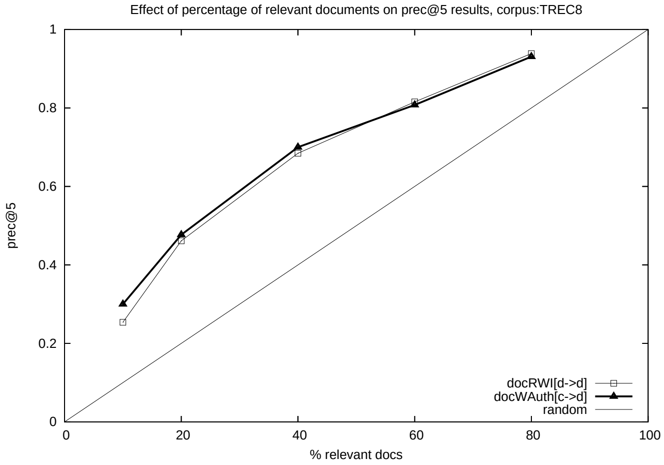
To further explore the connection between centrality (as induced by either doc-WAuth[c→d] or doc-RWI[d↔d]) and relevance we use the experimental setup from Section 6.3.4 to control the percentage of relevant documents in $\mathcal{D}_{\text{init}}$. We present the effect of this percentage on the prec@5 performance results of centrality-based ranking in Figure 7.2.

Figure 7.2: Centrality and relevance: the effect of varying the percentage of relevant documents in $\mathcal{D}_{\text{init}}$ on the performance (prec@5) of the doc-RWI[d↔d] and doc-WAuth[c→d] algorithms.

AP


Effect of percentage of relevant documents on prec@5 results, corpus:AP
prec@5
% relevant docs
0
0.2
0.4
0.6
0.8
1
0
20
40
60
80
100
docRWI[d->d]
docWAuth[c->d]
random

TREC8


Effect of percentage of relevant documents on prec@5 results, corpus:TREC8
prec@5
% relevant docs
0
0.2
0.4
0.6
0.8
1
0
20
40
60
80
100
docRWI[d->d]
docWAuth[c->d]
random

WSJ

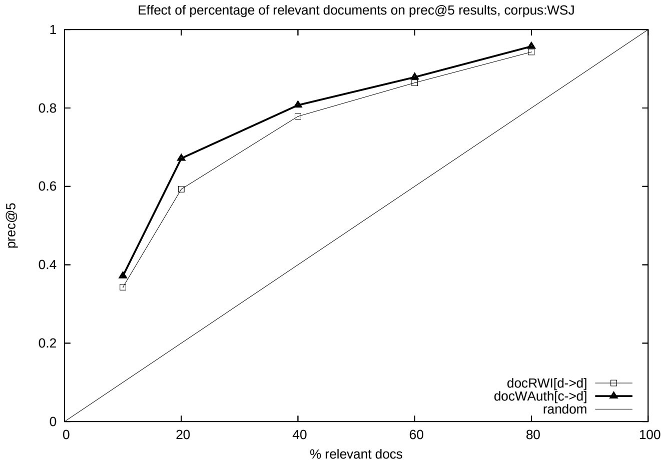

Effect of percentage of relevant documents on prec@5 results, corpus:WSJ
prec@5
% relevant docs
0
0.2
0.4
0.6
0.8
1
0
20
40
60
80
100
docRWI[d->d]
docWAuth[c->d]
random

Table 7.2: Main comparison: HITS or Recursive Influx on document-only graphs versus HITS on cluster-to-document graphs. Bold: best results per column. Symbols "*r*" and "*a*": doc-WAuth[c→d] result differs significantly from that of doc-RWI[d↔d] or doc-WAuth[d↔d], respectively. (For algorithms' names in the table, "doc-" is abbreviated as "d-".)

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
d-WAuth[d↔d]	.509	.486	.638	.440	.424	.648	.504	.464	.638
d-RWI[d↔d]	.519	.480	.632	.524	.446	.666	.536	.486	.699
d-WAuth[c→d]	**.541**	**.501** ^{r}	**.669** ^{r}	**.544** ^{a}	**.452**	**.674**	**.564** ^{a}	**.514** ^{a}	**.746** ^{a}

Table 7.3: Re-Ranking algorithms, as applied to either d↔d graphs or c→d graphs. Underline: best results for a given algorithm when the underlying graph is varied. Boldface: best results per column. Symbols "i" and "o": results differ significantly from the initial ranking or the optimized baseline, respectively. (For algorithms' names in the table, "doc-" is abbreviated as "d-".)

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
init. ranking	.457	.432	.596	.500	.456	.691	.536	.484	.748
opt. baselines	.465	.439	.635	.512	**.464**	**.696**	.560	.494	**.772**
d-WI[d↔d]	.515	.487 ^{io}	.643	.488	.432	.637	.520	.470	.644 ^{o}
d-WI[c→d]	.537 ^{io}	**.502** ^{io}	**.673**	.536	.436	.666	.564	.512	.704
d-RWI[d↔d]	.519	.480	.632	.524	.446	.666	.536	.486	.699
d-RWI[c→d]	**.543** ^{io}	.488 ^{io}	.649	.532	.432	.675	**.568**	.512	.704
d-WAuth[d↔d]	.509	.486	.638	.440	.424	.648	.504	.464	.638 ^{o}
d-WAuth[c→d]	.541 ^{io}	.501 ^{io}	.669	**.544**	.452	.674	.564	**.514**	.746

We can see in Figure 7.2 that the performance of doc-WAuth[c→d] (for almost all evaluation points on all corpora) is at least as good as — and on WSJ in general better than — that of doc-RWI[d↔d], attesting to the effectiveness of implementing HITS on the cluster-doc graphs. Furthermore, centrality as induced by both algorithms is connected with relevance, as the corresponding curves are above the line representing a random ranking of documents in $\mathcal{D}_{\text{init}}$.

Full suite of comparisons We now turn to Table 7.3, which gives the results for the re-ranking algorithms doc-WI, doc-RWI and doc-WAuth as applied to either the document-based graph d↔d (as in Chapter 6) or the cluster-document graph c→d. (Discussion of doc-WHub is deferred to Section 7.4.3.)

To focus our discussion, it is useful to first point out that in almost all of our nine evaluation settings (3 corpora × 3 evaluation measures), all three of the re-ranking algorithms perform better when applied to c→d graphs than to d↔d graphs, as the number of underlined numbers appearing in "d↔d" rows in Table 7.3 indicates. [9] Since it is thus clearly useful to incorporate cluster-based information, we will now mainly concentrate on c→d-based algorithms.

[9] This cannot be entirely attributed to having an extra cluster-size parameter:

Table 7.4: Cluster-based re-ranking. Bold: best results per column. Symbols i, o, c: results differ significantly from the initial ranking, optimized baseline, or (for the re-ranking algorithms) clust-$p_c(q)$ (Liu and Croft, 2004), respectively. (For algorithms' names in the table, "c-" stands for "clust-".)

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
init. ranking	.457	.432	.596	.500	.456	.691	.536	.484	.748
opt. baselines	.465	.439	.635	.512	.464	.696	.560	.494	**.772**
c-$p_c(q)$	.448	.418	.549 ^{io}	.500	.432	**.723**	.504 ^{o}	.454 ^{io}	.680
c-WI[d→c]	.511 ^{c}	**.479** ^{c}	.619	.524	**.478**	.681	**.568** ^{c}	**.512** ^{c}	.760
c-RWI[d→c]	.493	.475 ^{c}	.595	.496	.444	.683	.528	.490 ^{c}	.736
c-WAuth[d→c]	**.533** ^{ioc}	.478 ^{c}	**.651** ^{c}	**.532**	.460	.714	.552	.478	.757

The results for prec@5, the metric for which the re-ranking algorithms' parameters were optimized, show that *all* c→d-based algorithms outperform the prec@5-optimized baseline — significantly so for the AP corpus — even though applied to a sub-optimally-ranked initial set. (Although we recall that while the initial ranking is always inferior to the corresponding optimized baseline, the differences are never significant.) In contrast, the use of d↔d graphs never leads to significantly superior prec@5 results.

We also observe in Table 7.3 that the doc-WAuth[c→d] algorithm is always either the best of the c→d-based algorithms or clearly competitive with the best. Furthermore, pairwise comparison of it to each of the doc-WI[c→d] and doc-RWI[c→d] algorithms favors the HITS-style doc-WAuth[c→d] algorithm in a majority of the evaluation settings.

We also experimented with a few alternate graph-construction methods, such as sum-normalizing the weights of edges out of nodes, and found that the doc-WAuth[c→d] algorithm remained superior to doc-WI[c→d] and doc-RWI[c→d]. (The results are omitted since the precise numerical comparison does not yield additional information.)

All in all, these findings lead us to believe that not only is it useful to incorporate information from clusters, but it can be more effective to do so in a way reflecting the mutually-reinforcing nature of clusters and documents, as the HITS algorithm does.

7.4.2 Re-ranking by cluster centrality

We now consider the alternative, mentioned in Section 7.2.3, of using the centrality scores for *clusters* as an *indirect* means of ranking documents, in the sense of identifying clusters that contain a high percentage of relevant documents. Note that the problem of automatically identifying such clusters in the re-ranking setting has been acknowledged to be a hard task for some time (Willett, 1985).

in the case of PageRank, the damping factor λ is tunable in the d↔d case but, according to Theorem 1, irrelevant in the c→d case.

Nevertheless, as stated in Section 7.2.3, we experimented with Liu and Croft's general clusters-for-selection approach (Liu and Croft, 2004): rank the clusters, then rank the documents within each cluster by $p_d(q)$. Our baseline algorithm, clust-$p_c(q)$, adopts Liu and Croft's specific proposal of the *CQL* algorithm [10] — except that we employ overlapping rather than hard clusters — wherein clusters are ranked by the *query likelihood* $p_c(q)$ instead of one of our centrality scores.

Table 7.4 presents the performance results. Our first observation is that the clust-WI[d→c] and clust-WAuth[d→c] algorithms are superior in a majority of the relevant comparisons to the initial ranking, the optimized baselines, and the clust-$p_c(q)$ algorithm, where the performance differences with the latter sometimes achieve significance.

However, the performance of the document-centrality-based algorithm doc-WAuth[c→d] is better in a majority of the evaluation settings than that of any of the cluster-centrality-based algorithms. On the other hand, it is possible that the latter methods could be improved by a better technique for within-cluster ranking.

To compare the effectiveness of clust-WI[d→c] and clust-WAuth[d→c] to that of clust-$p_c(q)$ in detecting clusters with a high percentage of relevant documents — thereby neutralizing within-cluster ranking effects — we present in Table 7.5 the percent of documents in the highest ranked cluster that are relevant. (Cluster size (k) was fixed to either 5 or 10 and out-degree (α) was chosen to optimize the above percentage.) Indeed, these results clearly show that our best cluster-based algorithms are much better than clust-$p_c(q)$ in detecting clusters containing a high percentage of relevant documents, in most cases to a significant degree. However, there is still a substantial difference between the (average) percentage of relevant documents in the *optimal cluster*, that is, the cluster with highest percentage of relevant documents, and the (average) percentage in the cluster ranked highest by our methods.

7.4.3 Further analysis

Authorities versus hubs So far in this chapter, we have only considered utilizing the authority scores that the HITS algorithm produces. We recall from Section 6.3.7, that in document-only graphs (i.e., d↔d), authority scores yielded in general performance results superior to those resulting from utilizing hub scores.

The chart below shows the relative performance results on our bipartite cluster-doc graphs when utilizing either auth or hub scores. Specifically, the "documents?" column compares doc-WAuth[c→d] (i.e., ranking documents by authoritativeness) to doc-WHub[d→c] (i.e., ranking documents by hubness); similarly, the "clusters?" column compares clust-WAuth[d→c] to clust-WHub[c→d]. Each entry depicts, in descending order of performance (except for the one indicated tie) as one moves

[10] Note that the algorithms from Chapter 5 rank documents rather than clusters, as opposed to the CQL algorithm, and therefore the latter is a more appropriate choice to serve as a reference comparison for cluster-based ranking.

Table 7.5: Average relevant-document percentage within the top-ranked cluster. Optimal cluster: cluster with highest percentage. k: cluster size. Bold: best result obtained by the three different cluster-ranking approaches. c: result (of one of our algorithms) differs significantly from that of clust-$p_c(q)$, used in Liu and Croft (2004).

Cluster ranking	AP		TREC8		WSJ	
	k=5	k=10	k=5	k=10	k=5	k=10
optimal cluster	79.6	70.1	83.6	70.6	81.5	70.4
clust-$p_c(q)$	39.2	38.8	39.6	40.6	44.0	37.0
clust-WI[d→c]	48.7 ^{c}	**47.6** ^{c}	48.0	43.8	51.2 ^{c}	48.0 ^{c}
clust-WAuth[d→c]	**49.5** ^{c}	47.2 ^{c}	**50.8** ^{c}	**46.6**	**53.6** ^{c}	**49.0** ^{c}

left to right, those centrality scoring functions that lead to an improvement over the initial ranking: A stands for "authority" and H for "hub". Cases in which the improvement is significant are marked with a '*'.

		When do we improve the initial ranking by measuring the centrality of:	
		documents?	clusters?
AP	prec @5	A^*H	A^*H
	prec @10	A^*H	AH
	MRR	AH	A
TREC8	prec @5	AH	AH
	prec @10		HA
	MRR	H	H^*A
WSJ	prec @5	AH	AH (tie)
	prec @10	AH	H
	MRR		HA

We see that in many cases, hub-based re-ranking does yield better performance than the initial ranking. But authority-based re-ranking appears to be an even better choice overall, as was the case for document-only graphs.

HITS on "smoothed" one-way bipartite graphs In Section 6.3.7 we saw that employing HITS on document-only graphs that were smoothed using a PageRank approach (see Definition 5) helped to improve performance, although the resultant performance was in general still inferior to that of the Recursive Weighted Influx algorithm.

We now examine the effect of smoothing the c→d graph weight function on the performance of the doc-WAuth[c→d] algorithm. We employ a PageRank style smoothing technique but take care to preserve the one-way bipartite structure of c→d. Thus, we smooth the edge weight function $wt^{\mathrm{c}\rightarrow\mathrm{d}}(v_1 \rightarrow v_2)$ as follows:

Table 7.6: Comparison of performance results for the doc-WAuth[d↔d] and doc-WAuth[c→d] algorithms when run on graphs with (S) and without (U) edge-weight smoothing. For each algorithm, corpus, and evaluation measure, underline indicates which of (S) and (U) results in a better performance. Boldface marks the best performance for an evaluation setting (corpus × evaluation measure).

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
WAuth[d↔d] (U)	.509	$\underline{.486}$	.638	.440	.424	$\underline{.648}$	.504	.464	.638
WAuth[d↔d] (S)	$\underline{.511}$	.471	$\underline{.642}$	$\underline{.488}$	$\underline{.440}$	.637	$\underline{.520}$	$\underline{.470}$	$\underline{.642}$
WAuth[c→d] (U)	**$\underline{.541}$**	.501	**$\underline{.669}$**	**$\underline{.544}$**	**$\underline{.452}$**	**$\underline{.674}$**	.564	**$\underline{.514}$**	**$\underline{.746}$**
WAuth[c→d] (S)	**$\underline{.541}$**	**$\underline{.504}$**	.668	**$\underline{.544}$**	.434	.655	**$\underline{.568}$**	**$\underline{.514}$**	.711

Table 7.7: Performance results for using centrality as a sole criterion for ranking (doc-RWI[d↔d] and doc-WAuth[c→d]), or as a "bias" on query likelihood using Equation 6.3 (doc-RWI[d↔d]+LM and doc-WAuth[c→d]+LM). Bold: best result per column.

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
RWI[d↔d]	.519	.480	.632	.524	.446	.666	.536	.486	.699
RWI[d↔d]+LM	.531	.492	.630	.560	.460	.676	**.572**	.496	.747
WAuth[c→d]	**.541**	**.501**	**.669**	.544	.452	.674	.564	**.514**	.746
WAuth[c→d]+LM	.537	.493	.630	**.572**	**.490**	**.702**	**.572**	.510	**.771**

$$wt^{\mathrm{c}\rightarrow\mathrm{d}}(v_1 \rightarrow v_2; \lambda) \stackrel{def}{=} \begin{cases} \frac{1-\lambda}{|V_{\mathrm{Right}}|} + \lambda \frac{wt^{[\mathrm{c}\rightarrow\mathrm{d}]}(v_1 \rightarrow v_2)}{\sum_x wt^{[v_2]}(v_1 \rightarrow x)} & \text{if } v_1 \in Cl(\mathcal{D}_{\mathrm{init}}) \text{ and} \\ & \quad v_2 \in TopGen(v_1 \,|\, \alpha, \mathcal{D}_{\mathrm{init}}); \\ 0 & \text{otherwise.} \end{cases}$$

Table 7.6 presents results for the doc-WAuth[d↔d] and doc-WAuth[c→d] algorithms when run either on smoothed graphs (S) or unsmoothed graphs (U). (The results for doc-WAuth[d↔d] are those presented in Table 6.4 of Section 6.3.7.) Our first observation is that in most of the evaluation settings, smoothing of the c→d graph actually hurts the performance of the doc-WAuth[c→d] algorithm, contrary to the case for document-only graphs wherein smoothing d↔d helps to improve the performance of doc-WAuth[d↔d] as was first described in Section 6.3.7. However, even with a smoothed underlying graph, the doc-WAuth[d↔d] algorithm is still inferior in all cases to the doc-WAuth[c→d] algorithm implemented on an unsmoothed graph.

Thus, while we have shown throughout this chapter that information about document-cluster similarity relationships is very valuable, the results just mentioned suggest that such information is more useful in "raw" form.

Re-anchoring to the query In the previous chapter we showed that centrality scores induced over document-only graphs can be used as a multiplicative weight on document query-likelihood terms, the intent being to cope with cases in which centrality in $\mathcal{D}_{\text{init}}$ and relevance are not strongly correlated. (Refer back to Section 6.2.3.)

The same modification could be applied to the c→d-based algorithms, although it is not particularly well-motivated in the HITS case as mentioned in Section 6.3.7. While Recursive Influx correspond to a stationary distribution that could be loosely interpreted as a "prior", in which case multiplicative combination with query likelihood is sensible, it is not usual to assign a probabilistic interpretation to hub or authority scores.

Nonetheless, for the sake of comparison completeness, we applied this idea to the doc-WAuth[c→d] algorithm. We denote the resultant ranking algorithm as **doc-WAuth[c→d]+LM**, following our practice in Chapter 6 to use "+LM" for indicating that a centrality score serves as "bias" on query likelihood. (See Equation 6.3.) We report the resultant performance in Table 7.7 in comparison to that of the doc-RWI[d↔d] algorithm and its implementation as a "bias" on query-likelihood — **doc-RWI[d↔d]+LM** . (Observe that doc-RWI[d↔d]+LM is the Recursive Weighted Influx+LM algorithm from Chapter 6, following the abbreviation conventions presented in Section 7.2; see also Table 7.1 for all graph-based re-ranking algorithms).

We see in Table 7.7 that while the performance of doc-WAuth[c→d] degrades when it serves as a "bias" (i.e., the doc-WAuth[c→d]+LM algorithm) on the AP corpus, on WSJ and TREC8 the performance actually improves. Furthermore, in 8 of 9 relevant comparison settings (3 corpora × 3 evaluation settings) doc-WAuth[c→d]+LM outperforms doc-RWI[d↔d]+LM.

Thus, it may be the case that centrality scores induced over a document-based graph are more effective as a multiplicative bias on query-likelihood than as direct representations of relevance in $\mathcal{D}_{\text{init}}$ (see Chapter 6); but, modulo the caveat above, it seems that when centrality is induced over cluster-based one-way bipartite graphs, the correlation with relevance is much stronger, and hence this kind of centrality serves as a better "bias" on query-likelihood.

7.5 Related work

Ideas similar to ours by virtue of leveraging the mutual reinforcement of entities of different types, or using bipartite graphs of such entities for clustering (rather than using clusters), are abundant (e.g., Karov and Edelman (1998), Dhillon (2001), and Beeferman and Berger (2000)). Karov and Edelman (1998) model the mutual reinforcement of context and words for the task of word sense disambiguation. Dhillon (2001) uses bipartite graphs of documents and terms for producing clusters of documents and corresponding clusters of terms. Beeferman and Berger (2000) utilize bipartite graphs of queries and URLs to induce a clustering of each (i.e., clusters of URLs and clusters of queries).

Random walks (with early stopping) over bipartite graphs of terms and documents were used for query expansion (Lafferty and Zhai, 2001), but in contrast to our work, no stationary solution was sought. A similar "short chain" approach utilizing bipartite graphs of clusters and documents for ranking an entire corpus was recently proposed (Kurland et al., 2005), thereby constituting the work most resembling ours. However, again, a stationary distribution was not sought. Also, *query drift* prevention mechanisms were required to obtain good performance; in our re-ranking setting, we need not employ such mechanisms.

Chapter 8
Structural re-ranking: performance-comparison summary

Throughout the previous three chapters we focused on the re-ranking setting, wherein an initially retrieved list of documents ($\mathcal{D}_{\text{init}}$) is re-ranked to obtain high precision at top ranks. Alternatively, one can use information from the list to re-rank all documents in a corpus; this approach is the basis of pseudo feedback (PF) methods (Ruthven and Lalmas, 2003).

To compare these two paradigms, we contrast the performance of our best performing structural re-ranking algorithms from Chapters 5, 6 and 7 with that of state-of-the-art pseudo feedback methods. We follow Section 4.4.4 and use Rocchio's method (Rocchio, 1971) (as applied to the highest ranking documents) — which as originally formulated operates in a vector space — and the relevance model (Lavrenko and Croft, 2001), which is a state-of-the-art language-model-based approach to retrieval.

To set all methods compared on an equal footing, we use all documents in $\mathcal{D}_{\text{init}}$ to perform feedback with both PF methods, as these documents are re-ordered by our methods. Note that this implies that both Rocchio's method and the relevance model use (as "feedback") documents that are retrieved by a basic LM approach wherein the smoothing parameter is chosen to optimize average non-interpolated precision at 1000 (i.e., MAP).[1]

We found that using log(tf).log(idf) weights to represent both the query and the documents resulted in better performance for Rocchio's method than that attained by using tf.log(idf) weights (Zobel and Moffat, 1998). Furthermore, for both Rocchio's method and the relevance model we experimented with both the original approaches and with variants for which we employed term clipping, i.e., models that use only the terms with highest weight (likelihood). (Refer back to Section 4.4.4 for more details.) The number of terms for both models was chosen from: $\{5, 10, \ldots, 50, 100, \ldots, 600\} \cup \{\infty\}$, where ∞ means that no term-clipping was performed. The interpolation parameter for the relevance model was set to $\{0.1, \ldots, 0.9\}$, and the (positive) weighting coefficient for Rocchio's method was set to $\{0.01, 0.05, 0.1, \ldots, 0.9, 1, 2, 4\}$.

All parameters (of both the PF methods and of our methods) were chosen to optimize prec@5 following the optimization procedure described in Section 5.3.2.

As representatives of our structural re-ranking approaches we chose two groups of algorithms. The first consists of algorithms that use similarities within $\mathcal{D}_{\text{init}}$ without additional "re-anchoring to the query" information (in the form of integrating query-likelihood information). This group includes the aspect-f, doc-RWI[d↔d]

[1] Experiments with Rocchio's method wherein the initial search was instead performed using a basic vector-space approach yielded results inferior to those we present.

Table 8.1: Performance results of structural re-ranking algorithms from Chapters 5, 6 and 7 compared to those of Rocchio's method and the relevance model. Bold: best performance per column; "i": performance significantly better than that of the initial ranking. (The prefix "doc-" was omitted from the graph-based algorithms' names (doc-RWI[d↔d], doc-WAuth[c→d], doc-RWI[d↔d]+LM, doc-WAuth[c→d]+LM) due to formatting considerations.)

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
init. ranking	.457	.432	.596	.500	.456	.691	.536	.484	.748
aspect-f	*.537*	*.498*	*.628*	*.560*	**.504**	**.714**	.576	.504	.759
RWI[d↔d]	*.519*	*.480*	*.632*	*.524*	*.446*	*.666*	.536	.486	.699
WAuth[c→d]	**.541** ^{i}	**.501** ^{i}	**.669**	*.544*	*.452*	*.674*	.564	*.514*	.746
interpolation-f	*.537* ^{i}	*.498* ^{i}	*.628*	**.576** ^{i}	*.496*	*.687*	.592 ^{i}	.508	.767
RWI[d↔d]+LM	*.531* ^{i}	*.492* ^{i}	*.630*	*.560*	*.460*	*.676*	.572 ^{i}	.496	.747
WAuth[c→d]+LM	*.537* ^{i}	*.493* ^{i}	*.630*	*.572* ^{i}	*.490*	*.702*	.572	.510	.771
Rocchio	.481	.449	.589	.388 ^{i}	.336 ^{i}	.588	**.620**	.512	**.807**
Rel. Model	*.521* ^{i}	**.501** ^{i}	*.621*	*.540*	*.478*	*.694*	.588	**.524**	.748

(i.e., Recursive Weighted Influx) and doc-WAuth[c→d] algorithms from Chapters 5, 6 and 7 respectively. The second group is algorithms that do incorporate query-likelihood as an additional source of information for the scoring function of the first three algorithms (either via interpolation or as multiplicative bias using Equation 6.3): interpolation-f, doc-RWI[d↔d]+LM (Recursive Weighted Influx+LM) and doc-WAuth[c→d]+LM from Chapters 5, 6 and 7 respectively.

We note that both the six structural re-ranking algorithms mentioned above and the two PF methods serving as reference comparisons have two free parameters. Thus, combined with the fact that all algorithms use the exact same documents as the "ground set", we feel that the following performance comparison is as "fair" as possible.

Table 8.1 presents the performance results of our structural re-ranking algorithms when compared to those of the initial ranking and the PF methods (Rocchio and relevance model). Our first observation is that the best prec@5 results — the metric for which we optimized performance — are obtained in two out of the three corpora by one of our structural re-ranking algorithms. Furthermore, the interpolation-f algorithm posts prec@5 results that are better in almost all cases than those of both PF methods (except for Rocchio's method results on WSJ).

Another observation from Table 8.1 is that our algorithms from the first group, i.e., those that do not utilize query-likelihood information as an additional source of information (on top of the basic scoring function) post quite satisfying performance results — in 5 out of the 9 relevant comparison settings (3 corpora × 3 evaluation measures) the best performance is obtained by one of these algorithms. Specifically, the doc-WAuth[c→d] algorithm, which ranks documents by their centrality as induced by the HITS algorithm over bipartite cluster-doc graphs, is clearly the best performing algorithm (of all algorithms presented in Table 8.1) on the AP corpus.

Table 8.2: Performance results of our most effective structural re-ranking algorithms, doc-WAuth[c→d]+LM (the prefix "doc-" was omitted from the corresponding table entry) and interpolation-f, compared to those of versions of Rocchio's method and the relevance model wherein only $\mathcal{D}_{\text{init}}$ is ranked, instead of the entire corpus. Bold: best performance per column; "i": performance significantly better than that of the initial ranking.

	AP			TREC8			WSJ		
	prec@5	prec@10	MRR	prec@5	prec@10	MRR	prec@5	prec@10	MRR
init. ranking	.457	.432	.596	.500	.456	.691	.536	.484	.748
WAuth[c→d]+LM	**.537**	*.493*	**.630**	*.572*	*.490*	.702	.572	.510	.771
interpolation-f	**.537** ^{i}	**.498** ^{i}	*.628*	**.576** ^{i}	**.496**	.687	.592 ^{i}	.508	.767
Rocchio	.509	.468	.604	.476	.408 ^{i}	**.722**	**.612** ^{i}	**.526**	**.810**
Rel. Model	*.525* ^{i}	*.495* ^{i}	**.630**	*.544*	*.486*	.696	.580	.514	.735

Consideration of pairwise algorithm comparisons in Table 8.1 reveals that the interpolation-f and doc-WAuth[c→d]+LM algorithms are the best performing ones — the former being the only algorithm that posts significant prec@5 improvements over the initial ranking on all three corpora. This finding not only supports our premise that structural re-ranking can be an highly effective paradigm for attaining high precision at top ranks (recall that the PF methods rank all documents in the corpus), but also that clusters of documents in the initially retrieved list can convey helpful information either when incorporated into the aspect-model framework (as in the interpolation-f algorithm), or when modeled in our relevance-flow graph-based framework (as in the doc-WAuth[c→d]+LM algorithm).

To complete the comparison between our best performing structural re-ranking methods and the pseudo-feedback models, we also evaluate the performance of the latter when applied only to documents in $\mathcal{D}_{\text{init}}$, rather than to all the documents in the corpus. Thus, both our methods and the PF methods (in this version) exploit information from $\mathcal{D}_{\text{init}}$ to re-rank only documents in $\mathcal{D}_{\text{init}}$. We present the performance results in Table 8.2.[2]

The first observation we draw from Table 8.2 is that in many cases the performance of the PF methods is better than that presented in Table 8.1 — especially for Rocchio's model on the TREC8 corpus — implying that in some settings it might be beneficial to apply PF methods to the initial list rather than to the entire corpus to obtain high precision at top ranks. However, there are some cases in which the original versions of the PF methods are superior to those applied only to $\mathcal{D}_{\text{init}}$.

In comparing our two best performing re-ranking algorithms (interpolation-f and doc-WAuth[c→d]+LM) to the PF methods as applied (only) to $\mathcal{D}_{\text{init}}$ (Table 8.2), we see similar performance patterns to those evident in Table 8.1, attest-

[2] We also implemented a version of Rocchio's model that operates in the LM space. However, the resultant prec@5 performance was consistently worst than that of the relevance model and therefore the results are omitted.

ing to the effectiveness of our algorithms — especially when pairwise algorithm comparisons are considered. Specifically, our interpolation-f algorithm seems to be in general more effective than both PF methods (except for Rocchio's model on WSJ). Indeed, on two (out of the three) corpora it posts the best prec@5 performance. Furthermore, it is clearly much more effective than both PF methods on the TREC8 corpus, which is the most heterogeneous corpus of the three corpora tested. Perhaps most importantly, interpolation-f is the only algorithm in Table 8.2 that posts significant prec@5 improvements over the initial ranking for all three corpora. (Recall that prec@5 is the evaluation metric for which we optimized performance.)

Chapter 9
Summary and Future Work

We have shown throughout this thesis that modeling inter-document similarities using language models can form the basis of effective algorithms for ad hoc retrieval.

One representation for inter-document similarities that we employed was document clusters. Our algorithmic framework, first designed for ranking all documents in a corpus and then adapted for re-ranking documents within an initially retrieved (short) list to obtain high precision among the few highest-ranked documents, leverages language models of (either query independent or query-dependent) clusters and documents to result in highly effective (re-)ranking algorithms.

We then presented a means for incorporating pairwise inter-document similarities within a graph-based framework. Specifically, we constructed graphs over documents in an initially retrieved list, wherein links are induced by asymmetric language-model-based inter-document similarities. Centrality induced over these graphs was shown to be connected with relevance and to be a much more effective "bias" on query-likelihood than previously non-structural re-ranking approaches are. Combining our two similarity representation approaches — clusters and links within graphs — in a bipartite cluster-document graph-based framework resulted in highly effective centrality-based re-ranking algorithms.

Our best performing algorithms for either ranking all documents in a corpus or for re-ranking an initial list are competitive with — and in many occasions superior to — state-of-the-art retrieval approaches.

Future directions One interesting research challenge emerging from our graph-based framework is the theoretical modeling of centrality and relevance under the same model. As shown in Chapters 6 and 7, these two concepts are correlated, but are also *not* independent as centrality is induced over a list of documents retrieved in response to a query.

Another important research challenge is the theoretical modeling of the "anchoring to the query" concept. We showed in Chapters 4 and 5 that interpolation with query-likelihood scores results in highly effective (re-)ranking algorithms. (Recall that this interpolation was a result of an estimation choice for the conditional probability $p(q|c, d)$.) Then, in Chapters 6 and 7 we saw that using query-likelihood scores as a multiplicative bias on centrality values results in effective methods for re-ranking. Deriving one unified framework in which interpolation could represent a (generative) mixture model and centrality is explained as an intermediate quantity correlated with relevance might be a step towards bringing together these two different approaches for query-anchoring.

A conceptually interesting challenge is to couple the algorithmic framework utilizing clusters created offline with the graph-based framework that we proposed for structural re-ranking. One potential point of connection is the fact that the top-ranked clusters are used in the algorithmic framework to provide query context at retrieval time — a conceptual analog of the initial list of documents upon which re-ranking is performed. Furthermore, the highest ranked clusters are in essence the "top generators" of the query (with respect to all offline clusters), and therefore one possible direction might be to define a graph containing all entities in the corpus (i.e., documents, clusters and the query), wherein query context is defined using the top-generators concept.

BIBLIOGRAPHY

Allan, J. (2003). HARD track overview in TREC 2003: High accuracy retrieval from documents. In *Proceedings of the Twelfth Text Retrieval Conference (TREC-12)*, pages 24–37.

Allan, J., Connell, M. E., Croft, W. B., Feng, F.-F., Fisher, D., and Li, X. (2000). INQUERY and TREC-9. In *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*, pages 551–562. NIST Special Publication 500-249.

Ando, R. K. and Lee, L. (2001). Iterative residual rescaling: An analysis and generalization of LSI. In *Proceedings of the 24th SIGIR*, pages 154–162.

Azar, Y., Fiat, A., Karlin, A., McSherry, F., and Saia, J. (2001). Spectral analysis of data. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*.

Azzopardi, L., Girolami, M., and van Rijsbergen, K. (2003). Investigating the relationship between language model preplexity and ir precision-recall measures. In *Proceedings of SIGIR*, pages 369–370. Poster.

Azzopardi, L., Girolami, M., and van Rijsbergen, K. (2004). Topic based language models for ad hoc information retrieval. In *Proceedings of International Conference on Neural Networks and IEEE International Conference on Fuzzy Systems*, pages 3281–3286.

Baeza-Yates, R. A. and Reibeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.

Baker, L. D. and McCallum, A. K. (1998). Distributional clustering of words for text classification. In *Proceedings of the 21st SIGIR*, pages 96–103.

Baliński, J. and Daniłowicz, C. (2005). Re-ranking method based on inter-document distances. *Information Processing and Management*, 41(4):759–775.

Barzilay, R. and Lapata, M. (2005). Collective content selection for concept-to-text generation. In *Proceedings of HLT/EMNLP*, pages 331–338.

Bast, H. and Majumdar, D. (2005). Why spectral retrieval works. In *Proceedings of SIGIR*, pages 11–18.

Beeferman, D. and Berger, A. L. (2000). Agglomerative clustering of a search engine query log. In *Proceedings of KDD*, pages 407–416.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396.

Berger, A. and Lafferty, J. (1999). Information retrieval as statistical translation. In *Proceedings of SIGIR*, pages 222–229.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference*, pages 107–117.

Brown, P. F., Della Pietra, V. J., deSouza, P. V., Lai, J. C., and Mercer, R. L. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Buckley, C. (2004). Why current IR engines fail. In *Proceedings of SIGIR*, pages 584–585. Poster.

Buckley, C. and Voorhees, E. M. (2000). Evaluating evaluation measure stability. In *Proceedings of SIGIR*, pages 33–40.

Cai, D. and He, X. (2005). Orthogonal locality preserving indexing. In *Proceedings of SIGIR*, pages 3–10.

Chen, S. F. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.

Clark, S. and Weir, D. (2002). Class-based probability estimation using a semantic hierarchy. *Computational Linguistics*.

Collins-Thompson, K. and Callan, J. (2005). Query expansion using random walk models. In *Proceedings of the fourteenth International Conference on Information and Knowledge Managment (CIKM)*.

Collins-Thompson, K., Callan, J., Terra, E., and Clarke, C. L. (2004). The effect of document retrieval quality on factoid question answering performance. In *Proceedings of SIGIR*, pages 574–575. Poster.

Connell, M., Feng, A., Kumaran, G., Raghavan, H., Shah, C., and Allan, J. (2004). UMass at TDT 2004. TDT2004 System Description.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley series in telecommunications. Wiley-Interscience, New York.

Crestani, F., Lalmas, M., Rijsbergen, C. J. V., and Campbell, I. (1998). "is this document relevant? ... probably": A servey of probabilistic models in information retrieval. *ACM Computing Surveys*, 30(4):528–552.

Croft, W. B. (1978). *Organizing and searching large files of documents*. PhD thesis, University of Cambridge.

Croft, W. B. (1980). A model of cluster searching based on classification. *Information Systems*, 5:189–195.

Croft, W. B. (1995). What do people want from information retrieval? *D-Lib magazine*.

Croft, W. B. and Harper, D. J. (1979). Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35(4):285–295. Reprinted in Karen Sparck Jones and Peter Willett, eds., *Readings in Information Retrieval*, Morgan Kaufmann, pp. 339–344, 1997.

Croft, W. B. and Lafferty, J., editors (2003). *Language Modeling for Information Retrieval*. Number 13 in Information Retrieval Book Series. Kluwer.

Cronen-Townsend, S., Zhou, Y., and Croft, W. B. (2002). Predicting query performance. In *Proceedings of SIGIR*, pages 299–306.

Cronen-Townsend, S., Zhou, Y., and Croft, W. B. (2004). A language modeling framework for selective query expansion. Technical Report IR-338, Center for Intelligent Information Retrieval, University of Massachusetts.

Cutting, D. R., Karger, D. R., Pedersen, J. O., and Tukey, J. W. (1992). Scatter/Gather: A cluster-based approach to browsing large document collections. In *15th Annual International SIGIR*, pages 318–329, Denmark.

Daniłowicz, C. and Baliński, J. (2000). Document ranking based upon Markov chains. *Information Processing and Management*, 41(4):759–775.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Dhillon, I. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD Conference*, pages 269–274.

Diaz, F. (2005). Regularizing ad hoc retrieval scores. In *Proceedings of the Fourteenth International Conference on Information and Knowledge Managment (CIKM)*, pages 672–679.

El-Hamdouchi, A. and Willett, P. (1986). Hierarchic document clustering using ward's method. In *Proceedings of SIGIR*, pages 149–156.

El-Hamdouchi, A. and Willett, P. (1987). Techniques for the measurement of clustering tendency in document retrieval systems. *Journal of Information Science*, 13:361–365.

El-Hamdouchi, A. and Willett, P. (1989). Comparison of hierarchic agglomerative clustering methods for document retrieval. *The Computer journal*, 32(3):220–227.

Erkan, G. (2006). Language model based document clustering using random walks. In *Proceedings of HLT/NAACL*.

Erkan, G. and Radev, D. R. (2004). LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Fang, H. and Zhai, C. (2005). An exploration of axiomatic approaches to information retrieval. In *Proceedings of SIGIR*, pages 480–487.

Fuhr, N. (1992). Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255.

Gao, G., Nie, J.-Y., and Bai, J. (2005). Integrating word relationships into language models. In *Proceedings of SIGIR*, pages 298–305.

Gao, J., Nie, J.-Y., Wu, G., and Cao, G. (2004). Dependence langauge model for information retrieval. In *Proceedings of SIGIR*, pages 170–177.

Garfield, E. (1972). Citation analysis as a tool in journal evaluation. *Science*, 178:471–479.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*. The Johns Hopkins University Press, third edition.

Grassmann, W. K., Taksar, M. I., and Heyman, D. P. (1985). Regenerative analysis and steady state distributions for Markov chains. *Operations Research*, 33(5):1107–1116.

Griffiths, A., Luckhurst, H. C., and Willett, P. (1986). Using interdocument similarity information in document retrieval systems. *Journal of the American Society for Information Science (JASIS)*, 37(1):3–11. Reprinted in Karen Sparck Jones and Peter Willett, eds., *Readings in Information Retrieval*, Morgan Kaufmann, pp. 365–373, 1997.

Grimmett, G. R. and Stirzaker, D. R. (2001). *Probability and Random Processes*. Oxford Science Publications, third edition.

Grossman, D. A. and Frieder, O. (1998). *Information Retrieval: Algorithms and Heuristics*. Kluwer.

Harman, D. and Buckley, C. (2004). The NRRC reliable information access (RIA) workshop. In *Proceedings of SIGIR*, pages 528–529. Poster.

Harman, D. and Voorhees, E. M. (1998). Overview of the seventh text retrieval conference (trec-7). In *Proceedings of of the Seventh Text Retrieval Conference (TREC-7)*, pages 1–24.

Hartigan, J. A. (1975). *Clustering Algorithms.* Wiley series in probability and mathematical statistics. Wiley-Interscience, New York.

Hatzivassiloglou, V. and McKeown, K. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the 35th ACL/8th EACL*, pages 174–181.

He, X., Cai, D., Liu, H., and Ma, W.-Y. (2004). Locality preserving indexing for document representation. In *Proceedings of SIGIR*, pages 96–103.

Hearst, M. A. and Pedersen, J. O. (1996). Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *Proceedings of SIGIR*.

Hiemstra, D. (2001). *Using language models for information retrieval.* PhD thesis, Centre for Telematics and Information Technology, University of Twente.

Hiemstra, D. (2002). Term-specific smoothing for the language modeling approach to information retrieval: The importance of a query term. In *Proceedings of SIGIR*.

Hiemstra, D. and Kraaij, W. (1999). Twenty-One at TREC7: Ad hoc and cross-language track. In *Proceedings of the Seventh Text Retrieval Conference (TREC-7)*, pages 227–238.

Hiemstra, D., Robertson, S., and Zaragoza, H. (2004). Parsimonious language models. In *Proceedings of SIGIR*, pages 178–185.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196.

Hofmann, T. and Puzicha, J. (1998). Unsupervised learning from dyadic data. Technical Report TR-98-042, International Computer Science Institute (ICSI).

Hu, X., Bandhakavi, S., and Zhai, C. (2003). Error analysis of difficult TREC topics. In *Proceedings of SIGIR*, pages 407–408. Poster.

Iyer, R. and Ostendorf, M. (1999). Modeling long distance dependence in language: Topic mixtures vs. dynamic cache models. *IEEE Transactions on Speech and Audio Processing*, 7(1):30–39.

Jardine, N. and van Rijsbergen, C. J. (1971). The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 7(5):217–240.

Jeffreys, H. (1946). An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 186(1007):453–461.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *Proceedings of ICML*, pages 290–297.

Karov, Y. and Edelman, S. (1998). Similarity-based word sense disambiguation. *Computational Linguistics*, 24(1):41–59.

Kleinberg, J. (1998). Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 668–677. Extended version in *Journal of the ACM, 46:604–632, 1999.*

Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46:604–632.

Kraaij, W. and Spitters, M. (2003). Language models for topic tracking: The importance of score normalization. In (Croft and Lafferty, 2003), chapter 5, pages 95–124.

Kraaij, W. and Westerveld, T. (2001). TNO-UT at TREC9: How different are web documents? In *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*, pages 665–671.

Kraaij, W., Westerveld, T., and Hiemstra, D. (2002). The importance of prior probabilities for entry page search. In *Proceedings of SIGIR*, pages 27–34.

Krovetz, R. and Croft, W. B. (1992). Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems*, 10(2):115–141.

Kurland, O. and Lee, L. (2004). Corpus structure, language models, and ad hoc information retrieval. In *Proceedings of SIGIR*, pages 194–201.

Kurland, O. and Lee, L. (2005). PageRank without hyperlinks: Structural reranking using links induced by language models. In *Proceedings of SIGIR*, pages 306–313.

Kurland, O. and Lee, L. (2006). Respect my authority! HITS without hyperlinks utilizing cluster-based language models. In *Proceedings of SIGIR*.

Kurland, O., Lee, L., and Domshlak, C. (2005). Better than the real thing? Iterative pseudo-query processing using cluster-based language models. In *Proceedings of SIGIR*, pages 19–26.

Lafferty, J. and Zhai, C. (2003). Probabilistic relevance models based on document and query generation. In (Croft and Lafferty, 2003), pages 1–10.

Lafferty, J. D. and Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR*, pages 111–119.

Langville, A. N. and Meyer, C. D. (2005). Deeper inside PageRank. *Internet Mathematics*.

Lavrenko, V. (2000). Localized smoothing of multinomial language models. Technical Report IR-222, Center for Intelligent Information Retrieval (CIIR), University of Massachusetts.

Lavrenko, V. (2004). *A Generative Theory of Relevance*. PhD thesis, University of Massachusetts Amherst.

Lavrenko, V., Allan, J., DeGuzman, E., LaFlamme, D., Pollard, V., and Thomas, S. (2002a). Relevance models for topic detection and tracking. In *Proceedings of the Human Language Technology Conference (HLT)*, pages 104–110.

Lavrenko, V., Choquette, M., and Croft, W. B. (2002b). Cross-lingual relevance models. In *Proceedings of SIGIR*, pages 175–182.

Lavrenko, V. and Croft, W. B. (2001). Relevance-based language models. In *Proceedings of SIGIR*, pages 120–127.

Lavrenko, V. and Croft, W. B. (2003). Relevance models in information retrieval. In (Croft and Lafferty, 2003), pages 11–56.

Lee, K.-S., Park, Y.-C., and Choi, K.-S. (2001). Re-ranking model based on document clusters. *Information Processing and Management*, 37(1):1–14.

Lee, L. and Pereira, F. (1999). Distributional similarity models: Clustering vs. nearest neighbors. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 33–40.

Leuski, A. (2001). Evaluating document clustering for interactive information retrieval. In *Proceedings of the Tenth International Conference on Information and Knowledge Managment (CIKM)*, pages 33–40.

Leuski, A. and Allan, J. (1998). Evaluating a visual navigation system for a digital library. In *Proceedings of the Second European conference on research and advanced technology for digital libraries (ECDL)*, pages 535–554.

Levow, G.-A. and Matveeva, I. (2004). University of Chicago at CLEF2004: Cross-language text and spoken document retrieval. In *Proceedings of CLEF*, pages 170–179.

Li, X. and Croft, W. B. (2003). Time-based language models. In *Proceedings of the 12th International Conference on Information and Knowledge Managment (CIKM)*, pages 469–475.

Liu, X. and Croft, W. B. (2004). Cluster-based retrieval using language models. In *Proceedings of SIGIR*, pages 186–193.

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

Matveeva, I. (2004). Text representation with the locality preserving projection algorithm for information retrieval task. Master's thesis, University of Chicago.

Metzler, D. (2005). Direct maximization of rank-based metrics. Technical Report IR-338425, Center for Intelligent Information Retrieval, University of Massachusetts.

Metzler, D. and Croft, W. B. (2005). A Markov random field model for term dependencies. In *Proceedings of SIGIR*.

Metzler, D., Diaz, F., Strohman, T., and Croft, W. B. (2005). Using mixtures of relevance models for query expansion. In *Proceedings of the Fourteenth Text Retrieval Conference (TREC)*.

Mihalcea, R. (2004). Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *The Companion Volume to the Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 170–173.

Mihalcea, R. and Tarau, P. (2004). TextRank: Bringing order into texts. In *Proceedings of EMNLP*, pages 404–411. Poster.

Miller, D. R. H., Leek, T., and Schwartz, R. M. (1999). A hidden Markov model information retrieval system. In *Proceedings of SIGIR*, pages 214–221.

Mitra, M., Singhal, A., and Buckley, C. (1998). Improving automatic query expansion. In *Proceedings of SIGIR*, pages 206–214.

Morgan, W., Greiff, W., and Henderson, J. (2004). Direct maximization of average precision by hill-climbing, with a comparison to a maximum entropy approach. Technical Report 04-0367, The MITRE Corporation.

Nallapati, R. (2004). Discriminative models for information retrieval. In *Proceedings of SIGIR*, pages 64–71.

Nallapati, R. and Allan, J. (2002). Capturing term dependencies using a language model bsed on sentence trees. In *Proceedings of the 11th International Conference on Information and Knowledge Managment (CIKM)*, pages 383–390.

Ng, A. Y., Zheng, A. X., and Jordan, M. I. (2001). Stable algorithms for link analysis. In *Proceedings of SIGIR*, pages 258–266.

Ng, K. (2000). A maximum likelihood ratio information retrieval model. In *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*, pages 483–492.

Ogilvie, P. (2000). Nearest neighbor smoothing of language models in IR. Unpublished.

Ogilvie, P. and Callan, J. (2003). Combining document representations for known item search. In *Proceedings of SIGIR*, pages 143–150.

Otterbacher, J., Erkan, G., and Radev, D. R. (2005). Using random walks for question-focused sentence retrieval. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 915–922.

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278.

Papadimitriou, C. H., Raghavan, P., Tamaki, H., and Vempala, S. (2000). Latent Semantic Indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, 61(2):217–235.

Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of English words. In *Proceedings of ACL*, pages 183–190.

Pinski, G. and Narin, F. (1976). Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Information Processing and Management*, 12:297–312.

Ponte, J. M. (1998). *Probabilistic Language Models for Topic Segmentation and Information Retrieval.* PhD thesis, University of Massachusetts, Amherst, Massachusetts.

Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of SIGIR*, pages 275–281.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137. Reprinted in Karen Sparck Jones and Peter Willett, eds., *Readings in Information Retrieval*, Morgan Kaufmann, 1997.

Preece, S. E. (1973). Clustering as an output option. In *Proceedings of the American Society for Information Science*, pages 189–190.

Robertson, S. E. (1977). The probability ranking principle in IR. *Journal of Documentation*, pages 294–304. Reprinted in K. Sparck Jones and P. Willett (eds), *Readings in Information Retrieval*, pp. 281–286, 1997.

Robertson, S. E. and Sparck Jones, K. (1976). Relevance weighting of search terms. *Journal of American society for Information Science*, 27(3):129–146.

Rocchio, J. J. (1971). Relevance feedback in information retrieval. In (Salton, 1971a), pages 313–323.

Rooth, M., Riezler, S., Prescher, D., Carroll, G., and Beil, F. (1999). Inducing a semantically annotated lexicon via EM-based clustering. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Rosenfeld, R. (2000). Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE*, 88(9).

Ruthven, I. and Lalmas, M. (2003). A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review*, 18(2):95–145.

Salton, G. (1968). *Automatic Information Organization and Retrieval*. McGraw-Hill computer science series. McGraw-Hill, New York.

Salton, G., editor (1971a). Prentice Hall.

Salton, G. (1971b). Cluster search strategies and the optimization of retrieval effectiveness. In (Salton, 1971a), pages 223–242.

Salton, G. and Buckley, C. (1988). On the use of spreading activation methods in automatic information retrieval. In *Proceedings of SIGIR*, pages 147–160.

Salton, J., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Sanderson, M. (1994). Word sense disambiguation and information retrieval. In *SIGIR 94*, pages 142–151, Dublin, Ireland.

Sanderson, M. and Zobel, J. (2005). Information retrieval system evaluation: effort, sensitivity and reliability. In *Proceedings of SIGIR*, pages 162–169.

Shah, C. and Croft, W. B. (2004). Evaluating high accuracy retrieval techniques. In *Proceedings of SIGIR*, pages 2–9.

Si, L., Jin, R., Callan, J., and Ogilvie, P. (2002). A language modeling framework for resource selection and results merging. In *Proceedings of the 11th International Conference on Information and Knowledge Managment (CIKM)*, pages 391–397.

Singhal, A., Buckley, C., and Mitra, M. (1996). Pivoted document length normalization. In *Proceedings of SIGIR*, pages 21–29.

Song, F. and Croft, W. B. (1999). A general language model for information retrieval (poster abstract). In *Proceedings of SIGIR*, pages 279–280.

Sparck Jones, K., Robertson, S., Hiemstra, D., and Zaragoza, H. (2003). Language modeling and relevance. In (Croft and Lafferty, 2003), pages 57–71.

Sparck Jones, K., Walker, S., and Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments - part 1. *Information Processing and Management*, 36(6):779–808.

Spink, A. and Jensen, B. J. (2004). A study of web search trends. *Webology*, 1(2).

Spitters, M. and Kraaij, W. (2001). TNO at TDT2001: Language model-based topic detection. In *Topic Detection and Tracking TDT Workshop*.

Srikanth, M. and Srihari, R. (2004). Biterm language models for document retrieval. In *Proceedings SIGIR*, pages 425–426. Poster.

Stewart, W. J. (1994). *Introduction to the numerical solution of Markov chains*. Princeton University Press.

Tao, T., Wang, X., Mei, Q., and Zhai, C. (2006). Language model information retrieval with document expansion. In *Proceedings of HLT/NAACL*.

Tao, T. and Zhai, C. (2004). A two-stage mixture model for pseudo feedback. In *Proceedings of the 27th SIGIR*, pages 486–487. Poster.

Teahan, W. J. and Harper, D. J. (2003). Using compression-based language models for text categorization. In (Croft and Lafferty, 2003), pages 141–165.

Tishby, N. and Slonim, N. (2000). Data clustering by Markovian relaxation and the information bottleneck method. In *Advances in Neural Information Processing Systems (NIPS) 14*, pages 640–646.

Tombros, A., Villa, R., and van Rijsbergen, C. (2002). The effectiveness of query-specific hierarchic clustering in information retrieval. *Information Processing and Management*, 38(4):559–582.

Toutanova, K., Manning, C. D., and Ng, A. Y. (2004). Learning random walk models for inducing word dependency distributions. In *Proceedings of the International Conference on Machine Learning*.

Turtle, H. R. and Croft, W. B. (1990). Inference networks for document retrieval. In *Proceedings of SIGIR*, pages 1–24.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths, second edition.

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

Voorhees, E. M. (1985). The cluster hypothesis revisited. In *Proceedings of SIGIR*, pages 188–196.

Voorhees, E. M. (1986). The efficiency of inverted index and cluster searches. In *Proceedings of SIGIR*, pages 164–174.

Voorhees, E. M. (2000). Variations in relevance judgments and the measure of retrieval effectiveness. *Information Processing and Management*, 36(5):697–716.

Voorhees, E. M. (2002). Overview of the TREC 2002 question answering track. In *The Eleventh Text Retrieval Conference TREC-11*, pages 115–123.

Voorhees, E. M. (2005). Overview of the TREC 2005 robust retrieval task. In *Proceedings of the Fourteenth Text Retrieval Conference (TREC)*.

Voorhees, E. M. and Harman, D. K., editors (2000). *The Eighth Text REtrieval Conference (TREC-8)*. NIST.

Wei, X. and Croft, W. B. (2006). LDA-based document models for ad-hoc retrieval. In *Proceedings of SIGIR*.

Willett, P. (1985). Query specific automatic document classification. *International Forum on Information and Documentation*, 10(2):28–32.

Xu, J. and Croft, W. B. (1996). Query expansion using local and global document analysis. In *Proceedings of SIGIR*, pages 4–11.

Xu, J. and Croft, W. B. (1999). Cluster-based language models for distributed retrieval. In *Proceedings of SIGIR*, pages 254–261.

Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd Annual SIGIR*, pages 42–49.

Yom-Tov, E., Fine, S., Carmel, D., and Darlow, A. (2005). Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *Proceedings of SIGIR*, pages 512–519.

Zamir, O. and Etzioni, O. (1998). Web document clustering: a feasibility demonstration. In *Proceedings of SIGIR*, pages 46–54.

Zaragoza, H., Hiemstra, D., and Tipping, M. (2003). Bayesian extension to the language model for ad hoc information retrieval. In *Proceedings of SIGIR*, pages 4–9.

Zhai, C. and Lafferty, J. (2002). Two-stage language models for information retrieval. In *Proceedings of SIGIR*, pages 49–56.

Zhai, C. and Lafferty, J. D. (2001a). Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of CIKM*, pages 403–410.

Zhai, C. and Lafferty, J. D. (2001b). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR*, pages 334–342.

Zhang, B., Li, H., Liu, Y., Ji, L., Xi, W., Fan, W., Chen, Z., and Ma, W.-Y. (2005). Improving web search results using affinity graph. In *Proceedings of SIGIR*, pages 504–511.

Zhang, Y., Callan, J., and Minka, T. (2002). Novelty and redundancy detection in adaptive filtering. In *Proceedings of SIGIR*, pages 81–88.

Zhu, X. J. (2005). *Semi-Supervised Learning with Graphs.* PhD thesis, Carnegie Mellon University.

Zobel, J. (1998). How reliable are the results of large-scale search engines? In *Proceedings of SIGIR*, pages 307–314.

Zobel, J. and Moffat, A. (1998). Exploring the similarity space. *ACM SIGIR forum*, 18(1):18–34.