Efficient and Effective Retrieval Using Higher-Order Proximity Models

A thesis submitted for the degree of
Doctor of Philosophy

Xiaolu Lu,
School of Science,
College of Science, Engineering and Health,
RMIT University,
Melbourne, Victoria, Australia.

Supervisors:
Dr. Jason Shane Culpepper
Prof. Alistair Moffat

April 17, 2018

Declaration

I certify that except where due acknowledgment has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Xiaolu Lu
School of Science
College of Science, Engineering and Health
RMIT University
April 17, 2018

Credits

Portions of the material in this thesis have previously appeared in the following refereed publications:

- X. Lu, A. Moffat, and J. S. Culpepper. How effective are proximity scores in term dependency models? In *Proc. Australasian Doc. Comp. Symp.*, pages 89–90, 2014
- X. Lu, A. Moffat, and J. S. Culpepper. On the cost of extracting proximity features for term-dependency models. In *Proc. Int. Conf. Information and Knowledge Management*, pages 293–302, 2015
- X. Lu, A. Moffat, and J. S. Culpepper. The effect of pooling and evaluation depth on IR metrics. *Information Retrieval J.*, 19(4):416–445, 2016
- X. Lu, A. Moffat, and J. S. Culpepper. Efficient and effective higher order proximity modeling. In *Proc. Int. Conf. Theory of Information Retrieval*, pages 21–30, 2016
- X. Lu, A. Moffat, and J. S. Culpepper. Modeling relevance as a function of retrieval rank. In *Proc. Asian Information Retrieval Societies Conf.*, pages 3–15, 2016
- X. Lu, A. Moffat, and J. S. Culpepper. Can deep effectiveness metrics be evaluated using shallow judgment pools? In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 35–44, 2017

Acknowledgments

I must first thank my supervisors, J. Shane Culpepper and Alistair Moffat, for their guidance and support throughout the candidature. I would also like to thank my parents for their support throughout my life, without whose reassurance I would not be able to finish this thesis. This work was supported by RMIT University and Australia Research Council.

Contents

List of Figures

List of Tables

Glossary

Notation	Description
D	Document
$G_*()$	Rank-level estimator
$I(Q')$	Optimal interval set of a subquery Q'
J	A set of judgments, a superscript $'$ represents a reduced judgment set
Q	A given keyword query, contains query term q
Q'_l	A subquery of length l
R	A set of relevant documents
S	A retrieval system
$gain_.(r_i)$	Mapped gain value of relevance score
$\mathbf{S}_{k\times n}$	n systems to depth k for a given topic
$\mathcal{C}$	Document collection
$\mathcal{I}_k$	The set of optimal intervals of a query $\|Q\| = k$
$\mathcal{Q}'_l$	A set of subqueries that have length l, which are derived from Q
$\mathsf{k}_{D,i}$	Rank of document D given by system S_i
$g1_\ell$	A background estimate vector of all documents output in the first level optimization, using the l-th rank-level estimator.
$g0_{*,m}$	The rank-level gain vector of a document D estimated by an m-th rank-level estimator.
$\sigma_{i,j}$	Order of S_i and S_j
d	Pooling depth, a superscript $'$ represents a reduced pool
$f_{t,D}$	Within document term frequency of a term t
f_t	Document frequency of a term t
k	In evaluation setting, it is the evaluation depth
r_i	Relevance value (of a document) at rank position i
t	A term
$W_{\mathrm{M}}(i)$	Weight at rank i used in the weighted-precision metrics
M	An evaluation metric

Abstract

Information Retrieval systems are widely used to retrieve documents that are relevant to a user's information need. Systems leveraging proximity heuristics to estimate the relevance of a document have shown to be effective. However, the computational cost of proximity-based models is rarely considered, which is an important concern over large-scale document collections. The large-scale collections also make collection-based evaluation challenging since only a small number of documents are judged given the limited budget. Effectiveness, efficiency and reliable evaluation are coherent components that should be considered when developing a good retrieval system. This thesis makes several contributions from the three aspects.

Many proximity-based retrieval models are effective, but it is also important to find efficient solutions to extract proximity features, especially for models using higher-order proximity statistics. We therefore propose a one-pass algorithm based on the PLANESWEEP approach. We demonstrate that the new one-pass algorithm reduces the cost of capturing a full dependency relation of a query, regardless of the input representations. Although our proposed methods can capture higher-ordered proximity features efficiently, the trade-offs between effectiveness and efficiency when using proximity-based models remains largely unexplored. We consider different variants of proximity statistics and demonstrate that using local proximity statistics can achieve an improved trade-off between effectiveness and efficiency.

Another important aspect in IR is reliable system comparisons. We conduct a series of experiments that explore the interaction between pooling and evaluation depth, interactions between evaluation metrics and evaluation depth and also correlations between two different evaluation metrics. We show that different evaluation configurations on large test collections, where only a limited number of relevance labels are available, can lead to different system comparison conclusions. We also demonstrate the pitfalls of choosing an arbitrary evaluation depth regardless of the metrics employed and the pooling depth of the test collections. Lastly, we provide suggestions on the evaluation configurations for the reliable comparisons of retrieval systems on large test collections.

On these large test collections, a shallow judgment pool may be employed as assumed budgets are often limited, which may lead to an imprecise evaluation of system performance, especially when a deep evaluation metric is used. We propose an estimation framework for estimating deep metric score on shallow judgment pools. With an initial shallow judgment pool, rank-level estimators are designed to estimate the effectiveness gain at each ranking. Based on the rank-level estimations, we propose an optimization framework to obtain a more precise score estimate.

CHAPTER 1

Introduction

1.1 Historical Background of Information Retrieval Systems

Despite many advances in web search engines over the years, reliably satisfying the information needs of users remains a fundamental research problem. The study of information retrieval didn't begin with the web, but has a long standing history dating back to 1950s [106], as depicted in Figure 1.1. The phrase *Information Retrieval* (IR) can have several different definitions, and its main goal is to help users locate information that can satisfy their information needs. One of IR's primary concerns is to deal with large amounts of unstructured data, where no clear structure was defined making it difficult for a computer to interpret, for example textual information. This is in contrast to traditional relational database systems where data often has a predefined schema. Although a *search* process is commonly used when referring to Information Retrieval, it contains a wide range of tasks beyond the search task, such as *filtering* and *question answering*. There are two questions that naturally arise from this process: how can we organize information, and how can we find the set of documents satisfying a user's needs.

The task of helping people to address their information needs was frequently performed with the help of professionals, such as librarians, paralegals and similar expert searchers [70]. Conventional approaches to managing the large collections of information arose from the field of librarianship, and a hierarchical subject classification mechanism using catalog cards was widely employed as the organizing approach [103]. However, arranging information using catalog cards was unable to handle the increasing amount of documents. Therefore an alternative method was proposed by Taube et al. [112], which was a term-based inverted list mechanism. The idea of indexing each term instead of a catalog cards was further improved and implemented as one of the basic representation in computer based IR systems, supporting different retrieval tasks. A detailed description of this technique is presented by Zobel and Moffat [135].

The task of locating and returning documents that satisfies the user requires the development of methods that can assess whether or not the current document is likely to satisfy the information need. One of the simplest methods to fulfill this task is to exactly match the data with a user's query in the document collections, which is known as *Boolean* retrieval; and it requires users

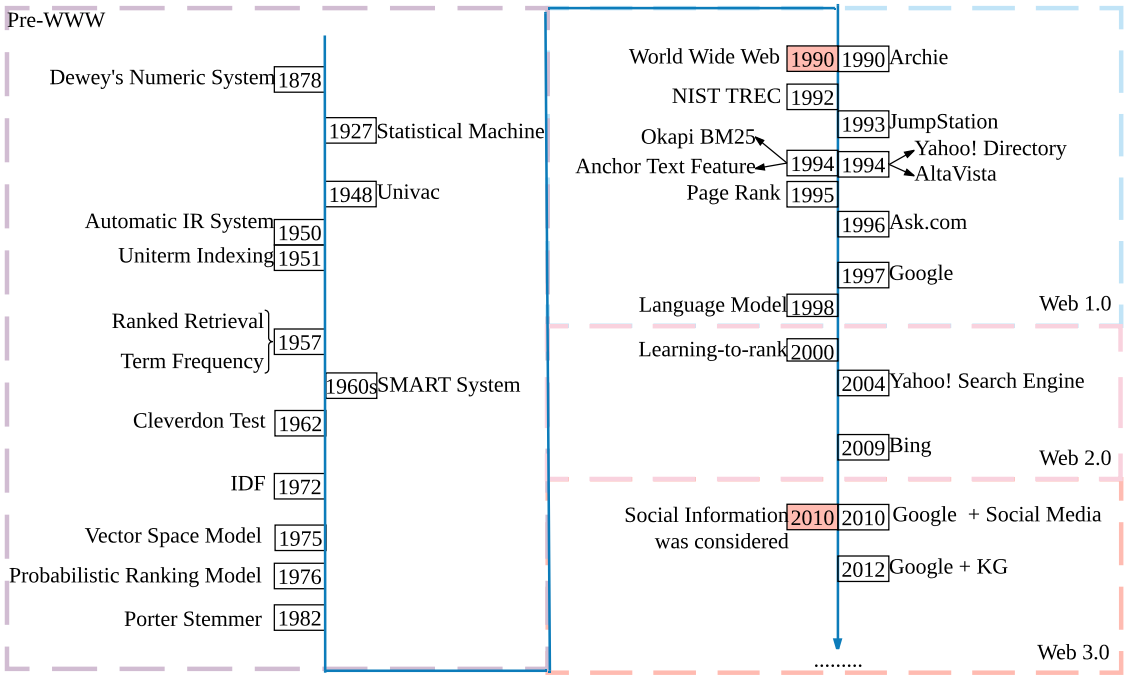


Figure 1.1: Timeline of information retrieval techniques and systems. The three dashed boxes contain key techniques and systems that appeared in different ages, categorized based on the birth of World Wide Web in 1990. The text to the left of each line are several techniques and the text to the right of the timeline are several key systems.

to express their information need in the form of a Boolean expression, using operators, such as "AND" or "OR". An alternative to Boolean retrieval is ranked retrieval, where no strict matching is required and the documents returned are in the form of a ranked list instead of a set. This method was widely adopted in the *ad hoc* retrieval process. Note that any retrieval systems aim to provide documents that are *relevant* to the information need of the user.

Most existing IR systems employ the ranked retrieval approach for providing users the set of desired documents. To do that, a similarity computation that estimates the likelihood of a document being relevant to the information need behind a query is required, which we refer to as a *retrieval model*. In 1960s, the vector space model was proposed [101], vectorizing each documents and computing cosine similarity between query and a document. A ranked list of documents sorted in descending cosine similarity which is then returned to the a user. The proposed *term frequency* (TF) and *inverse document frequency* (IDF) statistics in 1970s have become the most important statistics adopted in any retrieval model, even through until the present day. Robertson and Jones [87] provided further explanation of these two statistics from a probabilistic perspective. Meanwhile, the *probabilistic ranking principle* (PRP) was proposed by Robertson [84]. One of the most well-known retrieval models, the Best Matching 25 (BM25) model was proposed based on PRP [89], and it has become one of the foundational retrieval models in IR systems.

From 1990s until recently, the velocity and volume of data on the World Wide Web put forth many new challenges in the field of IR. This large-scale added new requirements to the challenges of developing retrieval systems. At the same time, the retrieval systems faced many scalability and

Figure 1.2: A snapshot of Google at late 1997 early 1998, obtained from http://blogoscoped.com/archive/2006-04-21-n63.html

efficiency challenges, which arose from not only the web-scale amount of data, but also the sophisticated computation required by modern retrieval methods. Moreover, the way that a retrieval system interacts with users also has been reevaluated and redesigned since the birth of Mosaic GUI Browser in 1993. Soon after, web search engines, such as Google emerged. Figure 1.2 shows a snapshot of Google in late 1997 and early 1998, where an input box was provided for users to input search queries and have the top-k results returned, where k was specified by the user. As hyperlinks and anchor text are important features for web pages, exploiting the two features in the retrieval model makes IR systems more effective, especially within the context of web search. Even today, more sophisticated retrieval models are being proposed, in order to better understand users' information needs thus provide high quality search results. Additionally, indexing and retrieval approaches have also attracted lots of attention from researchers, in order to devise retrieval systems that are scalable while at the same time providing results in an efficient manner.

1.2 Evaluating Retrieval Systems

Aside from designing better retrieval systems, evaluating system performance is also a crucial task in IR studies, since we need to answer the question: *Is system A better than system B?* as a guide for improving system performance. As shown in Figure 1.3, effectiveness scores obtained in the evaluation process can be used as a guide for improving existing retrieval models. Hence, reliable evaluation is an important component in the big picture of information retrieval studies. In order to compare two systems, an *evaluation metric* is required, which computes an *effectiveness score* for a retrieval system, as an indicator of system performance.

For Boolean retrieval systems, precision and recall is used for system evaluation, indicating the percentage of documents retrieved by a system that meets user's expectation; and among

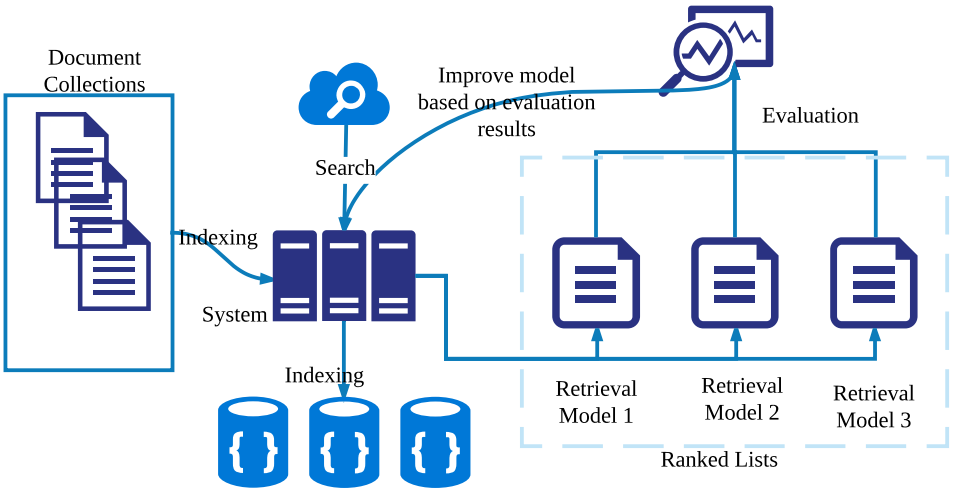


Figure 1.3: Relationship between evaluation and retrieval systems.

the set of documents that can satisfy user's information need, the percentage returned by the system. However, the two metrics are not suitable for ranked retrieval. In order to make use of ranked positions in the evaluation process, metrics such as *Average Precision* (AP) were proposed and became one of the standard measurements for assessing retrieval systems. The AP metric computes precision scores of a system, at all of the ranks at which documents can meet user's information needs; then averaged over the total number of relevant documents of that specific need.

The evaluation process can be performed either online or offline, and for offline evaluation, building resuable test collections is important for the evaluation task. The Text REtrieval Conference[1] (TREC) is a major annual exercise started in the 1990s, and which provides publicly available datasets for training and testing new retrieval systems. Not only in providing test collections, TREC also has a profound impact on developing new evaluation metrics. More importantly, approaches for evaluating quality of test collections have also been proposed during past conferences. The practice of defining search tasks and the methodology of building test collections adopted in TREC also give some guide to researchers for building their own offline test collections.

With the increasing size of the document collections, it is expensive to ask assessors to judge large amounts of documents. Therefore, cost-effective test collection building has received considerable attention. When using limited judgments for system evaluation, system comparison may be biased, thus results may not be reliable. Such bias may result from different evaluation configurations, consisting of metric and the maximum depth considered of a ranked list used for evaluation. In order to make a fair comparison between systems using the limited set of judgments, researchers propose different approaches, from the aspect of collection building and developing new evaluation metrics [51, 127]; from the aspect of configuring evaluation settings based on existing test collections [95]; and also from the aspect of estimating true performance of systems [55, 83, 121].

As a result of the large-scale of web data and also due to search engines becoming more

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

personal and contextual, evaluating using only the offline method is no longer sufficient, especially for commercial search engines. Hence, online evaluations such as A/B testing are adopted in practice, which measures system performance from direct observations of users.

As we mentioned, in IR studies, efficiency, effectiveness and evaluation are important components that are tightly connected with each other. In order to develop a high-performance retrieval system, it is essential that a reliable conclusion can be drawn from the evaluation process. Also, the trade-offs between effectiveness and efficiency should be carefully considered, since sophisticated retrieval models may result in expensive computations, which may be impractical. It is the space defined by these functions in which this thesis is presented, considering aspects of efficiency, of effectiveness trade-offs; and from the aspect of the reliable evaluation on large-scale test collections.

1.3 Thesis Structure

As we briefly introduced in the previous section, the field of Information Retrieval has a long history, including indexing techniques, retrieval models and evaluation methods. Chapter 2 gives an introductory view from all three aspects. We describe the background of retrieval models by considering the search process and retrieval heuristics. Firstly, we focus on the ranked retrieval process, and proceed to discuss the classic models that exist. More advanced features other than TF and IDF have also been leveraged in retrieval models, among which the closeness of terms in the documents has been considered as being one of the more effective techniques. We detail a range of proximity-based retrieval models and give a simple categorization of these models. The use of proximity improves retrieval effectiveness, but at increased computational cost. New indexing techniques have been proposed in order to perform a trade-off between effectiveness and efficiency. Since efficiency is crucial in the domain of a web search environment, given that the amount of documents in these collections is so large, accurately assessing the efficiency effectiveness trade-offs is of importance. Hence, we discuss the evaluation framework and issues of large-scale test collections. Chapter 2 closes with the description of the TREC-developed experimental materials heavily used in this thesis.

The contribution chapters begin in Chapter 3, where we propose approaches for computing proximity statistics for queries containing more than two terms. We consider the state-of-the-art term-dependency models proposed by Metzler and Croft [73], which consider the distance of queries of length greater than two, regardless of the order these query terms appear in the document. We firstly empirically examine the important role distance features play in the term-dependency models, compare the difference with phrase features, and reconfirm the effectiveness of proximity features. A new method is proposed for finding proximity statistics of all dependencies of a given query. By using a PLANESWEEP approach, it is possible to compute all dependencies within one pass of query evaluation, saving the computational overhead for the exponentially growing size of subqueries.

There are two factors that lead to the inefficiency of applying a term-dependency model in practice, the first is the exponential number of intervals with regards to query length, and the

second is the two-pass evaluation process of interval IDF values calculation. The second factor is considered in Chapter 4. When considering the entire query evaluation process of term-dependency model, the use of IDF in the computation process is costly. Computing collection-wise proximity statistics requires a two-pass evaluation over the entire index, which may be inefficient for some queries. Therefore, Chapter 4 assesses the trade-off between using collection-wise statistics and document-level statistics. Based on several retrieval heuristics and also alternative proximity-based models, we propose an alternative model that substitutes the global statistics with document-level statistics. We show that our proposed surrogates improve the effectiveness compared to bag-of-words models, but also reduce the computational overhead to some extent, when compared to models requiring global statistics.

Obtaining a reliable evaluation result is a vital step when developing new systems. We empirically show that on large test collections when only a limited volume of judgments are available, it is possible that biased results will be obtained, especially when the evaluation configuration is not appropriate. Two key components of the settings are the evaluation depth and the metrics used. Chapter 5 empirically examines the impact of different settings on the conclusions that can be made through this process. We show that, on the large TREC web test collections, where only top-ranked documents have usually been judged, a shallow weighted-precision metric may be the best choice for reaching a reliable evaluation conclusion.

When a deep evaluation is required, where using a weighted-precision metric also risks having lots of uncertainties in system comparisons, an approach that is able to estimate system effectiveness is required. Chapter 6 considers this problem and proposes a framework for estimating effectiveness scores when using deep weighted-precision metrics. The complete framework consists of two modules: a curve fitting component that models relevance as retrieval ranks, and an optimization framework that is dependent on the previous one, which outputs a unified estimate for each considered system.

The thesis ends with Chapter 7, where we summarize our findings, review their weaknesses and consider possible extensions.

CHAPTER 2

Background

2.1 Information Retrieval

Information Retrieval is an interdisciplinary field that is largely comprised of various tasks related to the storage, processing and searching of vast amounts of unstructured data such as textual documents, images, and video. A common search task involves a user formulating a query and receiving answers in the form of a summarized list of documents in ranked order [32]. Figure 2.1 outlines a typical search engine process. A user begins a search process with an *information need*, which is often represented by a set of *keywords*. These keywords are often referred to as *queries*. A retrieval system is designed to receive a query (Q) as input and calculate a similarity score between each document (D) in the corpus ($\mathcal{C}$) and the query. Let $Score(Q, D)$ be the similarity score between a query and a document; its aim is to measure the degree to which the resulting document is a representative candidate for reflecting the user's information need. As a final step of the retrieval process, a ranked list consisting of documents, sorted by descending similarity scores, is presented to the user. Normally, a user will reformulate the query if results are not satisfying.

One of the core components in a retrieval system is the *retrieval model*, which is concerned with computing the similarity score. A higher similarity score indicates that the document is predicted as being relevant to a user's information need. Retrieval models are discussed in detail in Section 2.2. In web search, *efficiency* is an important consideration, and an efficient retrieval system must present results to users in a reasonable response time by minimizing computational cost. A proper representation of documents in the document collection is crucial for achieving low query latency, which will be further detailed in Section 2.3. In order to *evaluate* the *effectiveness* of a retrieval system, various metrics have been proposed. These evaluation metrics measure to what extent the provided results satisfy users' information needs. Although there are different evaluation approaches, an offline evaluation is often employed in lab-based experimental environments. The offline evaluation process and related issues are presented in Section 2.4.

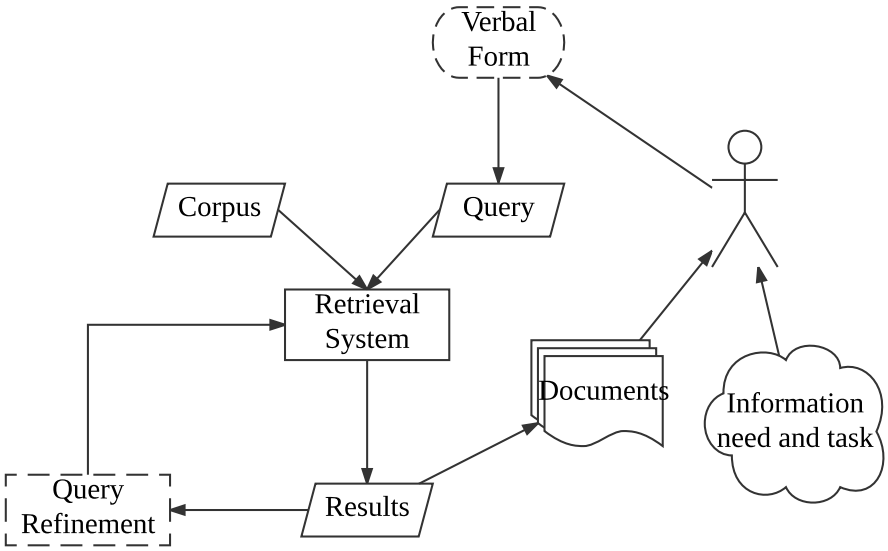


Figure 2.1: The whole retrieval task is completed by both users (dashed symbols) and retrieval systems (solid symbols). A user begins a retrieval task with an information need, which is then input into the retrieval system in the form of a *query*. Retrieval systems return a set of documents to the user from a document corpus.

2.2 Retrieval Heuristics and Models

A core component in every retrieval system is the one that is responsible for scoring the similarity between a document and a user query. Before ranked retrieval, Boolean retrieval was widely used. However, it has several drawbacks [106]. For example, it requires the input query in a well-formatted structure, which makes it cumbersome for inexperienced users. Therefore, free text queries have been widely adopted as the de facto input method for an information retrieval system. The standard approach for the presentation of search results is in the form of a ranked list which unequivocally conveys document importance. In order to provide a better result to satisfy complex information needs, ranked retrieval has become increasingly popular. In Section 2.2.1, basic retrieval models that calculate a similarity score based on unigrams, which are often referred to as *bag-of-words* (BOW) models, are discussed. Section 2.2.3 discusses more sophisticated ranking regime, which were proposed to improve user satisfaction. Finally, in Section 2.2.2, key factors and heuristics used in ranking models are discussed.

2.2.1 Basic Retrieval Models

Singhal [106] classified basic retrieval models into three types: the vector space model, probabilistic models, and inference models. In this section, three representative models from the first two classes are discussed: the vector space model, the BM25 retrieval model and the language model. The three models are also bag-of-words models that often serve as baselines in various

retrieval experiments. These bag-of-words models were developed based on term independence assumptions, which assume no dependencies between query terms and thus similarity scores are calculated for each query term individually.

Vector Space Model. One of the earliest ranking models was the vector space model proposed by Salton et al. [101]. It projects both query and documents into a vector space. Let Q be the query and D be a document, both of which have dimensionality equal to the number of unique terms in the corpus. Based on the vector representation, the similarity score is then calculated as:

$$\mathit{Score}(Q, D) = \frac{D \cdot Q}{\|Q\|_2 \|D\|_2}. \tag{2.1}$$

There are two options for a vector representation: (i) a binary representation indicating the presence or absence of each term, or (ii) a term weight ($w_{t,D}$) indicating the terms' importance in each document. The calculation of $w_{t,D}$ used in the second option usually follows a TF×IDF weighting scheme, which is based on term-frequency (TF) and inverse document frequency (IDF) [87]. Let $f_{t,D}$ be the frequency of a term t occurring in document D (TF), and let f_t be the number of documents containing term t, then one way of the TF×IDF weighting scheme is calculated as:

$$w_{t,D} = (1 + \log f_{t,D}) \cdot \log \frac{N}{f_t}, \tag{2.2}$$

in which N is the total number of documents in the document.

Using the binary representation in the vector space model may fail to differentiate between documents. For example, consider two documents D_1 and D_2, and two query term q_1 and q_2. If q_1 appears in D_1 and q_2 appears in D_2, both of which have a same TF values, but q_1 appears less often than q_2 in the entire document collection. Then, if a binary representation is used, D_1 and D_2 may have the same score, which means that they are estimated as being equally likely relevant to the query. However, when a weighting scheme such as Equation 2.2 is adopted, D_1 may be ranked higher since it captures additional information via the inverse document frequency that assigns a higher weight for the more rarely occurring term q_1.

The Probabilistic Ranking Model. Models in the second class are based on the probabilistic ranking principle (PRP), expressed by Robertson [84], as:

> "Documents should be ranked in such a way that the probability of the user being satisfied by any given rank position is a maximum".

As the probability cannot be directly obtained, models falling in this category mainly focus on estimating the probability using evidence from the documents.

One of the most widely used bag-of-words models in this category is Okapi BM25 [89], which scores documents using:

$$\mathit{Score}(Q, D) = \sum_{q \in Q} w_q \cdot \frac{(1 + k_1) \cdot f_{q,D}}{k_1((1 - b) + b(|D|/\mathit{avg\_}|D|)) + f_{q,D}} \cdot \frac{(1 + k_3) f_{q,Q}}{k_3 + f_{q,Q}}, \tag{2.3}$$

where $f_{q,D}$ is the within document TF value of a query term q, $f_{q,Q}$ is the occurrence of term q appearing in Q, and w_q is the IDF weighting and $avg\_|D|$ is the average document length. The k_3 parameter is useful when the query is long, but in retrieval tools such as Indri[1] or ATIRE[2], it is often omitted. There are several variants for calculating the IDF weight, and Robertson and Jones [87] discussed these alternatives. Among all variants, two are widely used:

$$\begin{aligned} w_t^{(1)} &= \log\left(\frac{N}{f_t}\right) \\ w_t^{(2)} &= \log\frac{N - f_t + 0.5}{f_t + 0.5}. \end{aligned} \tag{2.4}$$

The main difference between the two weighting schemes is that $w_t^{(1)}$ is negative when $f_t > N/2$, while $w_t^{(2)}$ tends to zero if f_t is close to N. Lee [54] compared different variants of IDF weighting, including the two listed in Equation 2.4. Lee argued that $w^{(2)}$ is more plausible, and a recent study by Trotman et al. [113] confirmed that conclusion. Many other BM25 variants have been proposed to improve the ranking model by adding heuristics such as document length, the importance of query terms, and so on. A detailed experimental study and survey on BM25 variants was provided by Trotman et al. [113], who concluded that, although there are many BM25 variants that consider more features, there is no model that consistently performs better than the others.

Another type of probabilistic model based on an inference network, was proposed by Turtle and Croft [114]. The network estimates the probability of a document being relevant to the query by using a variety of different features beyond unigrams.

The Language Model. Probabilistic models such as BM25 rank documents based on the probability that a document may be relevant to a query using statistical information derived from documents. A different idea is to calculate the probability that a document would generate the query, which is the core idea behind the language model (LM) [81]. One of the key components in any language model is smoothing, which is used to adjust the estimation error caused by data sparseness. Zhai and Lafferty [132] compared three smoothing methods: Jelinek-Mercer, Dirichlet and Absolute discount. Their experimental study showed that (i) retrieval performance is sensitive to the choice of smoothing method, and to the parameter settings; and (ii) different queries have different sensitivity patterns with regard to the smoothing methods. Moreover, Dirichlet performs well in general, and it is more suitable for retrieval tasks on web collections compared to Jelinek-Mercer, this can be attributed to the short query length common in such a scenario. Zhai and Lafferty also gave suggestions on the optimal parameter values used for each smoothing method, for example, $\mu = 2000$ should be considered when using Dirichlet smoothing. As a result, the Language Model with Dirichlet Smoothing (LMDS) is often used in retrieval systems. The model using Dirichlet smoothing will be regarded as the default language model in this thesis, which scores a document as follows:

$$Score(Q, D) = |Q| \cdot \frac{\mu}{|D| + \mu} + \sum_{q \in Q} f_{q,Q} \cdot \log\left(\frac{f_{q,D} \cdot |\mathcal{C}|}{\mu \cdot f_{q,\mathcal{C}}} + 1\right), \tag{2.5}$$

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

[2] http://atire.org

in which $f_{q,Q}$ is the frequency of a query term $q \in Q$, $|\mathcal{C}|$ is the total number of terms in the corpus, $f_{q,\mathcal{C}}$ is the frequency of q in the collection, which is $\sum_{D \in \mathcal{C}} f_{q,D}$.

Follow-up work has proposed a range of variants based on the original language model. In general, Zhai [131] concluded three aspects where these variants may vary compared to the original language model: (i) the assumption of the document model; (ii) the smoothing method; or (iii) the document prior, which will be defined shortly. Robertson and Zaragoza [88] compared language models to the BM25 framework on a theoretical level. Different document models capture assumptions of the probability of a term appearing in a document. For example, while both the multi-Bernoulli model and the multinomial model have an independence assumption of whether a term will appear in a document, they are slightly different. The first model assumes that each term occurrence is independent of other terms and the independence assumption adopted by the latter model takes the repeated occurrence into consideration. Beyond the two widely acknowledged models, recent studies show that other models such as the Poisson [71] model can also achieve promising results. As discussed, different smoothing methods have a large impact on retrieval effectiveness, thus a second class of language model variants can be obtained by choosing different smoothing methods. A *document prior* can incorporate static ranking preference of documents. For example, a basic language model can use a prior consisting of PageRank as a feature. Trotman et al. [113] empirically examine performance difference between BM25 variants and language model variants; both types of retrieval models have comparable effectiveness when correct parameter configurations are used.

2.2.2 Retrieval Heuristics and Features

In general, both probabilistic retrieval frameworks and language models have been successful in advancing the state-of-the-art of the retrieval process as they share a number of heuristics that are similar in nature. When improving upon the classic features, a great deal of attention has been on the study and proposal of more advanced statistical features and retrieval heuristics, especially in the learning-to-rank framework.

However, not all features are shown to improve retrieval effectiveness consistently. Therefore, in this section, retrieval heuristics used in classic models are presented, with a focus on proximity features. The last part of this section presents a simple overview of the other features used in more sophisticated ranking mechanisms.

Heuristic Retrieval Constraints. There are many factors that can lead to a successful retrieval model, such as TF and IDF weighting. Four categories of heuristics constraints have been formalized by Fang et al. [37] based on the basic retrieval models, including term frequency constraints (TFC), term discrimination constraints (TDC), length normalization constraints (LNC), and TF-length constraints (TF-LNC). The TFC are related to $f_{q,D}$ statistics, which include: (i) favoring a document with a higher $f_{q,D}$; (ii) favoring a document contain more distinct query terms; and (iii) preventing score increases caused by frequent terms. The TDC is a heuristic that calibrates between $f_{q,D}$ and f_q. The last two constraint types are related to document length, which aims at retrieving medium length documents and while penalizing long ones. All the constraints described

are included in popular retrieval models and have been shown to be key factors that improve retrieval effectiveness. Experimental results show that if a model fails in satisfying all of the listed constraints, then it is not optimal. Fang et al. showed that none of the retrieval models satisfy all constraints unconditionally, and thus, theoretically, they may be further improved.

Proximity Heuristics and Features. Besides the widely used TF×IDF heuristic, proximity features are one of the simplest methods used to extend bag-of-words models. One of the earliest studies on applying proximity features in ranked retrieval [42] stated the proximity hypothesis as:

> "The closer together a set of intersecting terms the more likely they are to indicate relevance".

Here, "interesting terms" may refer to query terms. For example, if a user is searching for "`scalable vector graphics`", the proximity heuristic suggests that if the query terms appear close to each other in one document more frequently, then this document is likely to be more relevant than another one which contains all terms, but are not clustered together.

There are various proximity statistical features that have been derived from such a heuristic, and they can be categorized using different aspects. The simplest categorization of these aspects that contribute the characteristic of proximity is a term-pair proximity feature that only considers two query terms. Higher-order proximity features exist, being more complex, they consider in addition to bigram relations, term sets of an arbitrary size. Consider the example query "`scalable vector graphics`". If there are no external resources available, it is hard for us to determine which combinations of two query terms are "interesting" but when considering all two-term combinations, of which the computation is expensive. Intuitively, a higher-order proximity feature, in this case, may be more effective since the query itself may be, for example, a location name and directly consider documents containing such phrases may produce higher quality results.

A second way to categorize proximity features is based on whether or not the interesting terms appear in the same order as the query. Requiring terms to appear in the same order as user queries is referred to as *sequential dependencies* [73]. The ordering constraint is often used with distance constraints, which requires the terms to appear consecutively. As an example, and while briefly ignoring the distance constraint, an ordered proximity feature requires "`scalable vector graphics`" to appear in exactly the same order in the document, but there may be other terms between the query terms as they appear within the document. For the case of an unordered proximity feature, there is no requirement for the query terms to appear in order within the document, still while omitting the consideration of any distance constraint.

The last class of proximity feature is classified based on the type of statistics used in capturing proximity heuristics – whether it is captured implicitly as TF×IDF statistics or a distance [111] measure. Let $Q' = \{\texttt{vector}, \texttt{graphics}\}$, and $I_{Q'} = \langle p_\ell, p_r \rangle$ be an interval in a document, which captures Q' located at position p_ℓ and p_r. Given a distance constraint, the first type of feature simply counts $f_{I,D}$ and f_I, which is the TF and IDF statistics of intervals satisfying the distance constraint. The second type of proximity feature is defined by aggregating interval lengths, $p_r - p_\ell + 1$. There are various options for such aggregation functions, and Tao and Zhai [111] have

explored several such variants, concluding that using the minimal distance as proximity features performs the best in their experimental studies.

Other Retrieval Features. In order to improve retrieval effectiveness, more sophisticated ranking approaches have been proposed, which involve features other than term proximity. Recent retrieval models, especially models in the learning-to-rank framework, make use of various features in order to produce a highly effective retrieval model. Macdonald et al. [68, 69] presented detailed experimental studies on the usefulness of features used in a learning-to-rank model. Although their experiments are in the context of learning-to-rank, combinations of these features explored by them have also been used by other models. For example, when considering a document structure, statistics from different document fields can also be used to give different weights in the BM25F model [90]. Macdonald et al. [68] showed that nearly all features can improve the retrieval effectiveness to some extent when used in a learning-to-rank model, but may result in decreased effectiveness when they are integrated into certain ranking models, for example, field based retrieval models.

2.2.3 Proximity-Based Retrieval Models

All of the models discussed in Section 2.2.1 make the assumption that the occurrence of query terms in a document are independent. As proximity is a simple concept that is relatively easy to integrate into existing bag-of-words models, as such there have been many studies on the development of proximity-based retrieval models. Table 2.1 presents an overview of the proximity-based retrieval models. In the table, proximity-based ranked retrieval models can be described roughly based on three key features: (i) whether the model contains higher-order proximity; (ii) whether the proximity is order constrained; and (iii) the type of statistical model. Based on how the proximity features are captured, representative models listed in the table will be discussed in the following paragraphs.

Frequency-Based Proximity Models. One of the most representative frequency-based models are the term-dependency models based on Markov Random Fields (MRFs) [73], which are implemented in Indri[3]. Metzler and Croft add proximity using a Markov Random Field, and then proposed two retrieval models – the Sequential Dependency Model (SDM) and the Full Dependency Model (FDM), both of which are derived from the LMDS family. In the modeling process, they consider three possible types of relationships among query terms: full independence (FI), sequential dependence (SD) and full dependence (FD).

Full independence is the same as the assumption adopted in the bag-of-words model, which is, the occurrence of each query term in a document is i.i.d. A *sequential dependence* assumes that the probability of a term appearing in a document will be affected by the adjacent query terms. Therefore, in SDM, bigram proximity is considered. If bigrams in the query occurr frequently in a document and within a fixed distance, then the document is considered to have a high probability

[3] https://www.lemurproject.org/indri.php

Model	Bigrams		Consec.		Statistics			Model Type		
	Y	N	Y	N	Freq.	Dist.	Kern.	BM25	LM	Other
Fagan [36] (1987)		✓	✓		✓					✓
Hawking and Thistlewaite [42] (1995)		✓		✓		✓				✓
Song and Croft [108] (1999)		✓	✓		✓				✓	
de Kretser and Moffat [34] (1999)	N/A	N/A	N/A	N/A			✓			✓
Clarke et al. [24] (2000)		✓		✓		✓				✓
Srikanth and Srihari [110] (2002)	✓		✓		✓				✓	
Rasolofo and Savoy [82] (2003)	✓			✓		✓		✓		
Metzler and Croft [73] (FDM) (2005)		✓		✓	✓				✓	
Metzler and Croft [73] (SDM) (2005)	✓			✓	✓			✓	✓	
Büttcher et al. [17] (2006)	✓					✓		✓		
Peng et al. [78] (SDM) (2007)	✓		✓		✓					✓
Peng et al. [78] (FDM) (2007)	✓			✓	✓					✓
Song et al. [109] (2008)		✓		✓		✓		✓		
Lv and Zhai [65] (2009)	N/A	N/A	N/A	N/A			✓		✓	
He et al. [43] (Window Counting) (2011)		✓	✓		✓			✓		
He et al. [43] (Survival Analysis) (2011)		✓	✓			✓	✓	✓		
Vuurens and de Vries [119] (2014)		✓		✓		✓			✓	

Table 2.1: Proximity-based ranked retrieval models and their characteristics. The "Bigrams" column shows whether the corresponding model using bigram features; the "Consec." column shows models impose order requirements on proximity features; the "Statistics" column indicates types of proximity statistics used in the model; the last column shows which bag-of-words the current proximity model is built on. A "✓" symbol shows the considered model satisfies the listed criteria; and the "N/A" means the model is not suitable for being categorized using the listed condition.

of being relevant to the query. Let $Q = \{q_1, q_2, \dots\}$ be the keyword query, $I_{ow}(Q')$ be a length w interval that captures a subquery Q' with order constraints, $I_{uw}(Q')$ be a length w interval that contains Q', without order constraints, and D be a document, then the canonical form of SDM can be written as:

$$\begin{aligned} \mathit{Score}(Q, D) =& \lambda_t \cdot \sum_{i=1}^{|Q|} \mathit{Score}(q_i, D) \\ &+ \lambda_{ow} \cdot \sum_{i=1}^{|Q-1|} \mathit{Score}(I_{ow}(\{q_i, q_{i+1}\}), D) \\ &+ \lambda_{uw} \cdot \sum_{i=1}^{|Q-1|} \mathit{Score}(I_{uw}(\{q_i, q_{i+1}\}), D)\,, \end{aligned} \tag{2.6}$$

where each function $\mathit{Score}(\cdot)$ is calculated in the same way as Equation 2.5, but using statistics of $I_{ow}(\{q_i, q_{i+1}\})$ or $I_{uw}(\{q_i, q_{i+1}\})$ instead of the unigram q_i. The difference between SDM and FDM is that FDM makes use of higher-order proximity features, which consider term-dependencies beyond bigrams. Therefore, computing FDM uses an equation similar to Equation 2.6, but instead of considering adjacent bigrams $Q' = \{q_i, q_{i+1}\}$, it considers all possible subqueries. Hence, the cost of computing FDM is exponential with regard to the length of Q, and can be expensive in practice. However, experimental studies show that on some query sets, FDM is more effective than other models, a result that is general for all queries and all collections.

Peng et al. [78] integrated term-dependencies into a Divergence From Randomness (DFR) [1] framework, which is implemented in the Terrier[4] system. Similar to FDM and SDM proposed by Metzler and Croft, both SD and FD variants have been considered, and the proximity heuristic is captured by counting the frequency across fixed length windows. However, different from the MRF-based FDM, only term-pairs are considered by Peng et al., and the full dependency only considered between two query terms.

There are also many other models in this category [36, 43, 108, 110]. Recent work by He et al. [43] proposes two different ways of using proximity features and the first method is to count n-gram frequency, counting the number of fixed windows that contain a given subquery, of which the core idea is similar to FDM; and the second method is to perform survival analysis to each window that contains query terms.

Distance-Based Proximity Models. Using frequency statistics requires a predefined distance constraint, which may miss important term intervals if the constraint is not selected properly [119]. Also, such fixed-length windows lose the information of exact distance among matched subqueries [43], which results in less discrimination ability between matched windows. Therefore, the second type of proximity models makes use of proximity heuristics by considering actual term distance within matched subqueries. Let I be an interval that captures an arbitrary subquery, and $\mathcal{I}$ be the set of intervals found in a document. Early work by Hawking and Thistlewaite [42]

[4] http://terrier.org/

consider two alternatives for calculating the contribution of intervals:

$$\begin{aligned} \mathit{Score}(\mathcal{I}, D) &= \sum_{I \in \mathcal{I}} \frac{1}{|I| - 1} \\ \mathit{Score}(\mathcal{I}, D) &= \sum_{I \in \mathcal{I}} \frac{1}{\sqrt{|I| - 1}}, \end{aligned} \tag{2.7}$$

where $|I|$ is the length of an interval. They show that on some topics, the retrieval effectiveness can be improved with the proximity features compared to a simple TF×IDF weighting approach. A slightly different approach of calculating the contributions of an interval was proposed in the cover density ranking model [24], where a predefined length w is required. Let I be an interval that contains an arbitrary subquery and w be the predefined fixed length, then its contribution is:

$$\mathit{Score}(I, D) = \begin{cases} w/|I|, & \text{if } |I| > w \\ 1, & \text{otherwise.} \end{cases} \tag{2.8}$$

The total contribution of the interval set $\mathcal{I}$ is the summation of individual intervals, which is similar to Equation 2.7.

Based on the BM25 retrieval model and also using the term proximity distance, Rasolofo and Savoy [82] and Büttcher et al. [17] proposed bigram based proximity using a BM25 statistical model. Both models score the contribution of an interval as:

$$\mathit{Score}(I, D) = \frac{1}{|I|^2}, \tag{2.9}$$

where $|I|$ is the length of the interval. When scoring the total contribution of the interval set, both models are within the BM25 framework and replace the TF statistics with interval distance weightings. A major difference between the two models is the way in which score aggregation over all matching intervals is performed. Rasolofo and Savoy [82] aggregate the score contributions by summing up contributions from all intervals, which is:

$$\mathit{Score}(\mathcal{I}, D) = \sum_{I \in \mathcal{I}} \mathit{Score}(I, D). \tag{2.10}$$

Büttcher et al. [17] amortize the total contribution of an interval to all query terms:

$$\mathit{Score}(\mathcal{I}, D) = \sum_{\substack{I \in \mathcal{I} \\ I \cap Q = \{q_i, q_j\}}} (w_i + w_j) \cdot \mathit{Score}(I, D), \tag{2.11}$$

where w_i and w_j are the IDF weightings of two matched query terms q_i and q_j, respectively. The work done by Song et al. [109] aligns these research lines, but generalizes to a higher-order proximity feature. Let I be the interval such that $I \cap Q \neq \varnothing$, for an individual interval, its contribution is calculated as:

$$\mathit{Score}(I, D) = \frac{|I \cap Q|^\lambda}{|I|^\gamma}, \tag{2.12}$$

in which γ and λ are two parameters and $|I|$ is the interval length. The main difference between Equation 2.12 and Equation 2.9 is that Equation 2.12 uses two parameters γ and λ when converting

distance to proximity statistics, which provides more flexibility. Equation 2.9 can be viewed as a special case of Equation 2.12, where $\lambda = 0$ and $\gamma = 2$. There are no conclusive results on which parameter should be applied in practice, and Song et al. [109] assigned the values empirically in their experiments, which is similar to Rasolofo and Savoy [82] and Büttcher et al. [17]. The final ranking model proposed by Song et al. [109] is computed in a similar way and was adopted by Büttcher et al. [17], where the contribution of an individual interval is amortized to query terms contained in the interval. The model proposed by Vuurens and de Vries [119] is based on KL-Divergence and the contribution of a single interval is:

$$Score(I, D) = \frac{|I \cap Q| - 1}{|I| - 1} \tag{2.13}$$

When compared to frequency-based methods, all models in this category can be computed without higher-order global features, and have a better discrimination even when two intervals have the same length. However, most models in this category consider term-pairs only, with the exception of the last two models [109, 119], which go beyond two-terms.

Kernel-Based Proximity Models. A third family of proximity models use proximity heuristics by defining different kernel functions, rather than explicitly using it with frequency counting or distance calculations. The main idea is to assume that the influence of each term is additive and the contribution changes based on the location within a document. de Kretser and Moffat [34] assumed four different options for kernel functions that can be used to describe how contributions of each term change based on its positions in a document. In a similar line of research, Lv and Zhai [65] proposed the Positional Language Model (PLM), which integrates different kernel functions into a language model. PLM is a position specific model, and each position in a document is given a different score. Therefore, three strategies of how to compute a final score for a document have been discussed: (i) use the maximum score of all positions (best position); (ii) average of all scores obtained in a document (multi-position); and (iii) computing a final score based on multiple propagation ranges (multi-σ). Among all of the combinations of design choices, Lv and Zhai have shown that a Gaussian kernel with multi-σ performs the best.

The survival analysis model proposed by He et al. [43] combines both kernel and distance-based methods. He et al. consider proximity contributions by modeling the probability of query term co-occurrences. While it is different from the kernel based method, the high-level idea is similar – model how proximity features are propagated in a document. In their experiments, variants of proposed models are compared, and also the effectiveness difference between models using the bigram feature and the trigram feature. Their experimental results show that using trigram features only marginally increase the retrieval effectiveness, and a frequency-based method is sufficient for improving the effectiveness since survival analysis performs similarly to counting fixed length windows.

Comparing Proximity-Based Models. Due to the extensive coverage of proximity-based retrieval models, one natural question is: *how do proximity-based retrieval models differ from each*

other with regards to retrieval effectiveness? A recent experimental study was conducted by Huston and Croft [45]. They mainly focus on comparing retrieval effectiveness between bigram and higher-order proximity models. They compare the term-pair model (BCTP) [17], the span-based term-pair model (TPSpan) [82], DFR-based proximity models (bDFR, pDFR), and variants of SDM and FDM models. They also considered PLM [65], however, due to excessive costs of this model, it was not applied to large corpora. There are two main conclusions drawn from their experiments – that higher-order proximity models cannot consistently outperform bigram models; that the performance of the retrieval models varies on queries, which result in similar performance on average among all variants. However, Huston and Croft also pointed out that the higher-order proximity models may potentially improve retrieval effectiveness, but requires a careful selection of dependent query terms.

Another comparison focus is between models that use global statistics and local statistics [66]. Models that require global statistics [73, 78] of proximity features have a higher computational cost since the computation can only be completed after traversing the entire index. Therefore, the scoring process of models that use statistics such as IDF need a two-pass algorithm in order to return the top-k documents. In contrast, models [17, 24, 43, 82] that use local proximity statistics only, such as the TF values of a fixed length window within a document, are less expensive. Besides the global statistics of unigrams, there are no requirements on the global statistics of proximity features, which means a lower retrieval cost when compared to models using global statistics. Macdonald and Ounis [66] studied the differences between the two types of models, FD and SD derived for both MRF and DFR retrieval frameworks. Macdonald and Ounis concluded that using global statistics of higher-order proximity may not lead to any significant improvement in the retrieval effectiveness, and when the queries get longer, it can result in performance degradation in both effectiveness and efficiency. Also, global statistics are not crucial even in large web corpora. Based on their empirical observations, it is possible to make use of local proximity features as a surrogate for the global ones, in order to reduce computational cost.

We consider both effectiveness and efficiency aspects in this work. Since making use of higher-order proximity features may improve the effectiveness and using local statistics may save some computational overhead when applying proximity features, the approach proposed in later chapters combines them.

2.3 Document Representations

In order to calculate similarity scores using the models discussed in Section 2.2, a proper representation of documents in the test collection is necessary. Before representing documents in the collection, there are several preprocessing steps required. Typically, each document in the collection is assigned a unique identifier (*doc_id*), and each term also has a unique identifier (*term_id*). A dictionary (or vocabulary) is built during the process of parsing all documents in the corpus. The most straightforward representation is to preserve each document in the form of term ID vectors, without any further processing, which is referred to as a *direct file*. However, an obvious drawback in using this representation is that it may not help improve the query evaluation cost in the ranked

D_1: `Scalable Vector Graphics is an XML-based vector image format.`

D_2: `The SVG Specification primarily focuses on vector graphics markup language.`

D_3: `Scalable Vector Graphics images can be produced by the use of a vector graphics editor.`

D_4: `All aspects of an SVG document can be accessed and manipulated using scripts in a similar way to HTML.`

Figure 2.2: An example document collection consisting four documents.

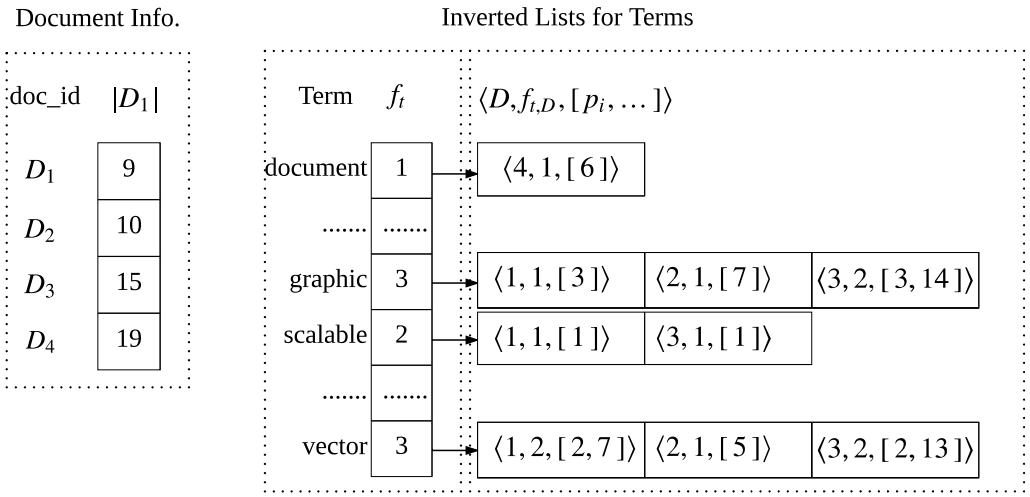


Figure 2.3: The basic inverted file structure for the example collection in Figure 2.2.

retrieval process. Consider the basic bag-of-words models discussed in Section 2.2.1. All of them fall in the TF×IDF ranking regime, and computing these statistics on-the-fly results in a high cost in retrieval process. Hence a representation that improves query evaluation is required. In this section, the main focus is the inverted file structure, including the one that is built for proximity ranking models. Some other representations are discussed in Section 2.3.2. After constructing a vocabulary, the four documents can be represented using a basic inverted index, as shown in Figure 2.3.

2.3.1 Inverted Lists

Basic Inverted Index. Consider an example document collection containing four documents, shown in Figure 2.2. A basic inverted file index structure contains two components: the vocabulary and an inverted list for each term [135]. The vocabulary contains all unique terms in the document collection, and the document frequency f_t associated with each term. Each term in the vocabulary also has a pointer reference to an inverted list that keeps the frequency statistics in the form of $\langle D, f_{t,D} \rangle$, which is the document ID where the term appears, and the number of occurrences of the term in document D. In addition to these two components, there is often a separate structure that maintains meta information for all documents. Besides a mapping from original document name to document ID, it may also contain information such as document length for the convenience of query evaluation. It is clear that, with the help of such an index structure, most of the bag-of-words

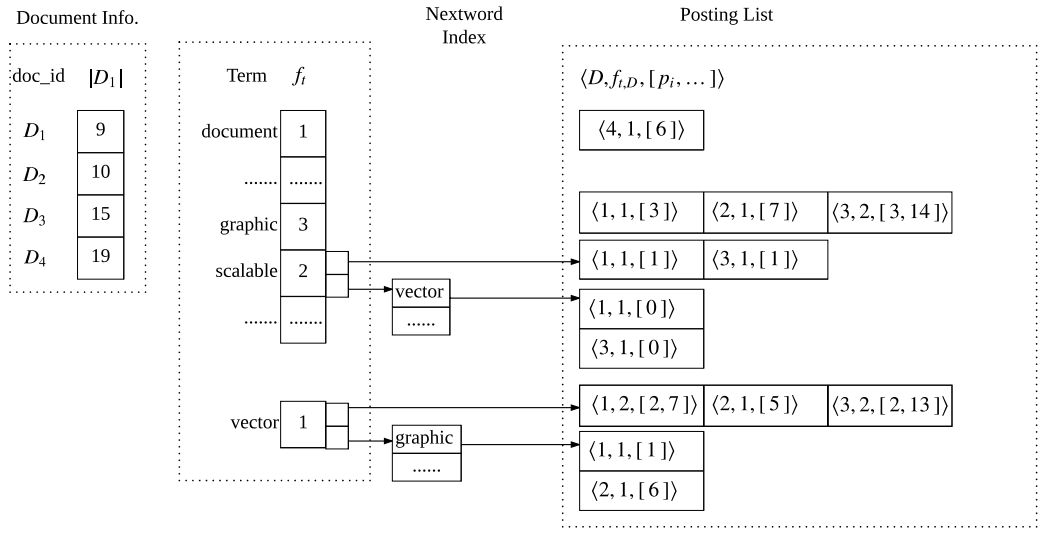


Figure 2.4: An example of a combined inverted list and partial nextword index.

models can be computed efficiently.

There are various ways of organizing such inverted lists. The inverted list of a term can be sorted based on document ID, but it can also be sorted based on impact or term frequency to improve the retrieval efficiency [2]. However, when the ranking model requires phrase or other proximity features, the basic inverted index is no longer sufficient. A natural extension is to add positional information to the inverted list of each term. Instead of keeping $f_{t,D}$ only, positions, where each term occurs, are also kept in the inverted list entry, which is the "$[p_i \ldots]$" part shown in Figure 2.3. Including positional information of terms adds to the space required and may not even be necessary if the retrieval process requires only document-level information. Zobel and Moffat [135] provide a detailed survey in regard to using inverted files for retrieval tasks.

Representing Proximity Statistics. As proximity features are query dependent, pre-computing and indexing can be difficult, especially for higher-order proximity features. However, if only phrases or term-pairs are considered in the proximity-based ranking models, it is possible to build a separate index structure. In addition to using the positional index only, there are three alternatives in representing phrase or term-pair proximity features – a hybrid index structure consisting both phrase and term-level inverted files [13, 125]; or an approximate index [35].

Phrase Index. A *phrase index* may also be stored using an inverted file representation where each entry in the vocabulary is a phrase instead of being a single term [125]. However, only representing a document collection using a phrase index is not sufficient since the term-level statistics are also required. Therefore, a commonly adopted solution is to use a hybrid index structure that consists of both term and phrase-level statistical information. The main concern with building a phrase-level index is the index size. Building an inverted index for all phrases is infeasible and unnecessary. Therefore, a *partial phrase index* is a viable alternative. A partial phrase index only

stores a set of common phrases, and is smaller compared to the alternative of storing everything. Needless to say, the subset may make an exact phrase match impossible, resulting in a failed query evaluation. To tackle this, Williams et al. [125] proposed the *nextword index*, which organizes terms in a tree-level structure and aims at optimizing retrieval of pairs of terms. It may still be infeasible to build a nextword index for all terms in the vocabulary.

Therefore, Williams et al. proposed to build the nextword index only for frequently occurring terms, as shown in Figure 2.4. For rarely used terms, only an inverted file is used in the query evaluation process. The final option of a hybrid index structure includes the partial phrase index in addition to an inverted file and a nextword index. Using a complete nextword index and an inverted file can help evaluate phrase queries efficiently, and a combination of a partial nextword index, a partial phrase index and an inverted file gives the most significant improvement in the evaluation of phrase queries with minimal space overhead. Also using a partial phrase index combined with an inverted index, Broschart and Schenkel [13] focused on improving the retrieval efficiency for the term-pair proximity model proposed by Büttcher et al. [17]. The partial term-pair index is no longer restricted to phrases according to the proximity features used in the retrieval model, so the number of possible entries in a term-pair index is much larger than only storing bigrams. The problem of pruning the size of the term-pair index has been formulated as an optimization problem by Broschart and Schenkel, where the index size is fixed and the optimization goal is to maximize the result quality. Broschart and Schenkel showed that, by using an effectiveness oriented hybrid index structure, retrieval effectiveness can be improved; using an efficiency oriented hybrid index, the efficiency of retrieval can be greatly improved with little effectiveness loss.

Approximated Representation of Proximity Statistics. While most of the research efforts have been put into building an auxiliary index structure that stores exact statistics of proximity features, Elsayed et al. [35] proposed an approximation approach. Based on SDM (Equation 2.6), Elsayed et al. explored the hypothesis – "By using approximated term positions, it is possible to obtain space and efficiency gains with little sacrifice in effectiveness". The core idea of the proposed approximation method is to divide ordinal positions of terms into buckets. Instead of storing the exact position lists for a term, only the IDs of buckets where a term appeared in are kept. There are two possible bucketing strategies: either fixing the bucket width or fixing the amount of buckets. Different bucket creation methods may impact both efficiency and effectiveness. Moreover, the compression mechanism used may also be an important factor in exploring the trade-offs between effectiveness and efficiency. In order to consider both bucketing and compression factors, Elsayed et al. experimented with all four combinations and concluded that creating buckets with a fixed width of 20 obtains the best trade-off between effectiveness and efficiency. Alternatively Huston et al. [46] proposed a sketch based index in order to support term-dependency models, which is more robust. The proposed index structure makes use of the CountMin sketch technique [31] in order to estimate n-gram frequency statistics. The main purpose of proposing a sketched index is to explore the space and time trade-offs when n-gram statistics have been applied in the retrieval models. In previous work, such as building a term-pair [13] or a hybrid phrase index [125], the vocabulary of the n-grams is often required to be stored as a part of index. While

0	a	8	editor	16	language	24	scripts	32	xml
1	all	9	focuses	17	manipulate	25	similar		
2	an	10	format	18	markup	26	specification		
3	aspect	11	graphic	19	of	27	svg		
4	assessed	12	html	20	on	28	the		
5	based	13	image	21	primarily	29	use		
6	be	14	in	22	produced	30	vector		
7	can	15	is	23	scalable	31	way		

Vocabulary

D_1	23	30	11	15	2	32	5	13	10	
D_2	28	27	26	21	9	20	30	11	18	16
D_3	23	30	11	13	7	6	22	...	8	
D_4	1	3	.....							

Direct File

Figure 2.5: An example of direct file representation of documents in Figure 2.2

the vocabulary size can be reduced by pruning infrequent phrases based on query logs, using the sketch method can avoid this step, because only hash values instead of actual terms are stored in the index structure. Huston et al. [46] empirically show that, by using the sketch technique, an optimal index structure with low space overhead and small errors can be constructed to support proximity-based models using n-gram features.

2.3.2 Representations Other than Inverted Lists

As briefly mentioned before, the naive and straightforward representation of a document is a direct file, which represents documents in a collection using a vector of term IDs, as shown in Figure 2.5. By traversing the term IDs, it is the same as reading the original document. Although this representation has been used less in the retrieval phase, it can be economical during the feature extraction stage. For example, Clarke et al. [24] extract and compute proximity statistics by concatenating all documents in the collection. Asadi and Lin [3] compare both representations for the feature extraction task and argued that a direct file is easier to extend when incorporating rich features such as named entity markups. Also, when a snippet generation task is required, keeping a direct file representation is crucial. A second choice for representing documents is using a self-index structure [33, 80]. Culpepper et al. proposed a self-index structure for retrieving documents based on bag-of-words retrieval models with high efficiency. Navarro and Mäkinen [77] gave a detailed introduction to self-indexing and its application in information retrieval.

Most work using self-indexing to represent documents can only support bag-of-words models, and few can be applied to term-dependency models. In order to support using term-dependency models to evaluate queries, Petri et al. [80] proposed a hybrid structure consisting an FM-INDEX structure and a pruned suffix tree structure. Unlike previous approaches that apply self-indexing

	Structure	Order	Ordered	Unordered	Approx.
Basic Index	Positional Inverted Index	Any	✓	✓	
Broschart and Schenkel [13]	Term-pair Inverted Index	Two	✓	✓	
Williams et al. [125]	Phrase Index	Any	✓		
Williams et al. [125]	Nextword Index	Any	✓		
Elsayed et al. [35]	Bucket-based Inverted Index	Two	✓	✓	✓
Huston et al. [47]	Sketch-based Inverted Index	n	✓		✓
Petri et al. [80]	Hybrid Self-index	n	✓		

Table 2.2: A summarization of representing proximity statistics.

for the index structure, the proposed hybrid approach gives exactly the same top-k documents as using exhaustive retrieval regime.

2.3.3 Summary of Representing Proximity Feature Statistics

Although pre-computing proximity statistics is difficult, especially when higher-order proximity features are considered, indexing phrases or term-pairs is a viable approach. Table 2.2 summarizes different approaches of pre-computing proximity-based statistics. A plain positional index can also support any types of proximity-based retrieval models, but with a more expensive query evaluation time cost compared to storing pre-computed proximity statistics.

However, supporting arbitrary proximity features is difficult, so most of the pre-computation focuses on a specific type of proximity feature. For example, when considering the order of proximity features, term-pair index structures have been proposed, regardless of whether the pairs are required to appear in an ordered window or not. When considering the higher-order proximity features, often ordered windows are used. While one of the main ideas is to avoid the space overhead, other trade-offs have been explored for these index structures. For example, the tunable index proposed by Broschart and Schenkel is based on finding an optimal point of effectiveness and space trade-off. Petri and Moffat [79] compare five different index structures for phrases, including using a direct file. Their results showed that the efficiency largely depends on two factors – the number of results from a given query and the smallest document frequency value among all query terms. Therefore, there is no indexing technique which is superior over all of the others, and more detailed studies on categorizing the proposed indexing structures are needed.

2.4 Test Collection-Based Evaluation

System evaluation is a broad topic in IR research, and can be viewed from several different perspectives. Broadly, studies can be categorized as being from a system or a user perspective [70]. In IR evaluation, researchers are particularly interested in measuring the *effectiveness* of retrieval systems [102], which is obviously assessed from a user perspective. The effectiveness of a retrieval system is defined as how well the returned results meet the information need of the user [70, 102, 115]. An ideal experiment to measure effectiveness requires a sufficiently large sample of actual users, which makes it expensive [115]. The test collection-based evaluation frame-

work has been commonly used since the Cranfield II [27] experiment, and provides a cost-effective alternative to real user studies. The paradigm was later adopted by many shared-task collaborative evaluations, including the Text REtrieval Conference (TREC)[5], the Cross-Language Evaluation Forum (CLEF)[6], and the NII-NACSIS Test Collection for IR Systems (NTCIR)[7] [102, 115]. The Cranfield paradigm also makes several assumptions, summarized by Voorhees [115]. In order to be able to evaluate systems reliably, collection-based evaluation requires a careful design. A test collection contains three important components [70, 102, 115]: (i) a set of topics representing users' information needs; (ii) a document collection; and (iii) for each topic, a set of judged documents indicating whether each is relevant to the information need. As shown in Figure 2.1, a retrieval system returns a ranked list of documents to satisfy a user's information need. Based on the results and judgments, an effectiveness score is calculated using an *evaluation metric* as an indicator of the system's performance. The focus of this section is on three main elements in the evaluation process. Section 2.4.1 introduces the evaluation framework with an emphasis on TREC-style test collection construction. Before proceeding to definitions of some widely adopted evaluation metrics and their differences, we first present the definitions on the relevance in 2.4.2. Section 2.4.3 includes several commonly used system and metric comparison methods over ideal judgments and then we will introduce the metrics developed when evaluating systems on the set of incomplete judgments in Section 2.4.4. After defining evaluation metrics, we introduce meta-evaluation approaches in Section 2.4.5. Finally, the experimental datasets used in this thesis are described in Section 2.5.

2.4.1 Judgment Set Selection

In principle, all documents in the collection should be judged, which is true for a few TREC early tasks. However, with the increasing size of document collections, it is too expensive to judge exhaustively for one collection and across a reasonable range of topics. Therefore, new strategies are needed in order to select the documents to be judged based on a judgment budget.

Shared Evaluation Tasks. Most of the widely used experimental test collections, including judgments, come from TREC tasks, which has been held since 1992. At the starting point of TREC, there were two tracks: an *Ad Hoc* task and a *Routing* task. In the Ad Hoc tracks, a set of topics are provided along with the document collection; systems are required to find a set of documents that are most relevant to each given topic. The routing track is different, where the set of information needs is static, but the documents are dynamic, and the goal of systems is to identify the set of documents similar to the ones provided [39]. Since 2002, additional tasks were added to TREC, each of which focuses on different challenges in various areas. There are domain specific tracks such as Genomics and Legal; also there are tracks that focus on enhancing different techniques, such as Spam and QA; in addition to textual based retrieval, content-based tracks were also introduced the video track [118]; moreover, the web [40] and TeraByte tracks [25] provide

[5] http://trec.nist.gov/
[6] http://www.clef-initiative.eu/
[7] http://research.nii.ac.jp/ntcir/index-en.html

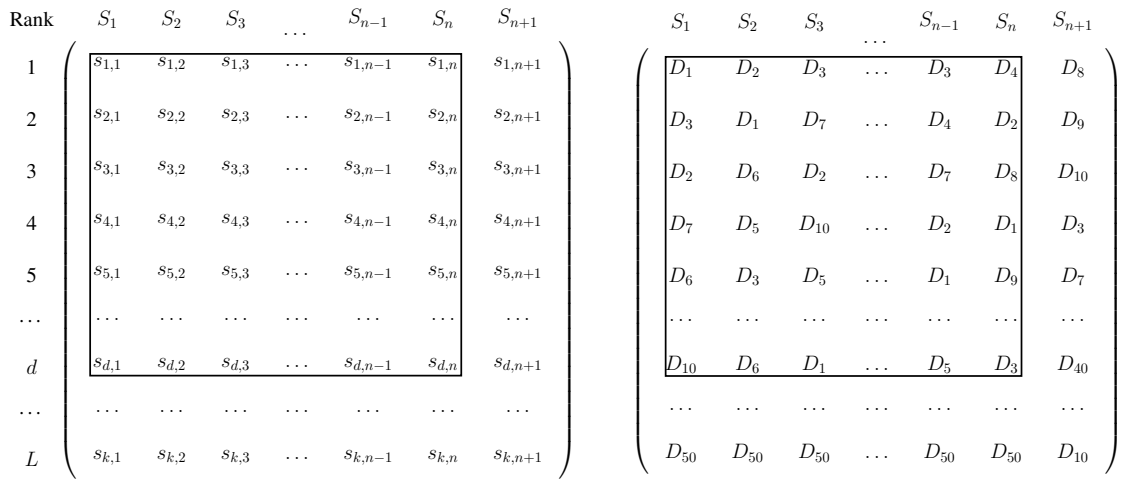


Figure 2.6: The top-d pooling process. Among all $n + 1$ participants, there are n contributing systems. Each row is a document vector returned by systems at rank k and each column represents the ranked list returned by a system.

the possibility for researchers to consider building large-scale retrieval systems that can be applied in World Wide Web (WWW) environment as a target. Among all of the tracks, we focus on the ad hoc retrieval tasks, where newswire or web-based document collections may be adopted. A detailed description of the topics and documents will be presented in Section 2.5. Other shared tasks such as NTCIR and CLEF also have a similar range of tracks, but may differ in focus, for example, more cross lingual tasks are involved. All shared tasks deliver publicly available test collections and TREC is the most influential of these and we will discuss the process of building test collections in the next several paragraphs.

The Fixed Depth Pooling Method. One solution to the limited budget problem is to make use of the ranked list submitted by contributing systems. Among all participants, we choose some of them as being *contributing* systems where the judged documents are sourced. Since only a limited labeling budget is available, we can only consider a subset of documents returned by contributing systems. By making use of the key aim of any IR system that the returned ranked lists are sorted in descending order of the likelihood of a document being relevant to the topic, the TREC top-d pooling method was proposed and is still in use today. Intuitively, if sufficient high quality IR systems are taken to be the sources of a judgment pool, most of the relevant documents are identified. This idea is core to the top-d pooling method, shown in Figure 2.6, where n systems are considered to be the source of a judgment pool. Each column in the left part of Figure 2.6 represents a ranked list submitted by one of the systems presumed to be part of the joint experiments. Documents falling in the rectangle are then selected as candidates to be judged. Note that although the figure labels systems in order, there is no such requirement in the original pooling process. In other words, documents from all systems are selected in parallel, regardless of which ranked list they appeared in, as is indicated by the rectangle. As d is the maximum rank of a document to be judged, it is likely to have a large impact on both the judgment cost and quality. A small d value may result in missing relevant documents while a large one may cost too much

judgment effort. In TREC NewsWire test collections, the pooling depth is 100, but for recent large document collections, the pooling depth is around 20.

Concerns have been raised regarding to the shallow pooling depths. One argument is that the small judgment pool can not identify the majority of relevant documents. If we assume the unjudged documents are not relevant, then there is a high risk that system performance can not be evaluated precisely. As pointed by Voorhees [116], organizers for other tasks, such as NTCIR, may use some additional manual runs to complement the pool. However, in general, the missing documents have little impact on early test collections such as TREC-5 [134]. Also, the *leave-one-out* experiments conducted by Zobel [134] showed that new systems such as S_{n+1} in Figure 2.6 can often be fairly evaluated. This is because a majority of relevant documents were found by the pooling process. That is, the assumption that unjudged documents are not relevant is reasonable in the early NewsWire test collections. Nonetheless, as is pointed out by Buckley et al. [16], the shallow judgment pool results in biased system comparisons, especially when a new system is evaluated. The trade-off between judgment budget and test collection quality has been referred to as the problem of building reusable test collections [21].

Sampling-Based Selection. A second option is to sample a set of documents to be judged from the candidate pool. One of the earliest work was done by Cormack et al. [29], who showed that it is possible to construct a judgment set using a sampling method to estimate the precision of a retrieval system. One simple and straightforward method is to perform uniform sampling with regard to the judgment budget [127]. However, this uniform sampling method ignores the fact that retrieval systems tend to put documents that are likely to be relevant in top ranked positions. Selecting documents uniformly may result in a judgment pool with few relevant documents and thus leading to biased estimations of system effectiveness. To mitigate this problem, stratified sampling is often employed in practice. For example, the test collection of TREC 2009 Ad Hoc task is constructed using stratified sampling [26], which has a shallow initial pooling depth $d = 12$. How different sampling strategies and how the design choices of strata affect the evaluation results is recently studied by Voorhees [117], considering the quality of different sampling methods by measuring the accuracy of the total number of relevant documents found. Under the guidance of a judgment budget and some evaluation metrics, more cost-effective collection construction methods have been studied [4, 5, 21]. Other than the budget benefits, these proposed methods are often tailored to one evaluation metric. Moreover, the exact extent to which they can provide unbiased evaluation results remains unclear.

Interactive Searching and Judging. While the top-d pooling method selects documents to be judged by treating documents in the $d \times n$ range (shown in Figure 2.6) equally, a further improvement may be to consider prioritizing documents that have higher estimated probabilities of being relevant. The intuition of prioritizing documents that are more likely to be relevant is often referred to as the *Move-to-Front* (MTF) heuristic. The *Interactive Searching and Judging* (ISJ) method introduced by Cormack et al. [28] makes use of MTF. It selects documents by considering the likelihood of relevance for all documents in the document collection. It is known that

documents appearing in each ranked list are sorted in descending order with respect to the probability of being relevant as estimated by that system, but how best to aggregate across multiple ranked lists remains challenging, because there is no indication of the quality of each ranked list. To compensate for this, Cormack et al. considered an estimate of the quality of each ranked list by using a "human-in-the-loop" approach. For each ranked list, a set of documents are judged according to their positions in the ranked list. Based on these judged documents, each system is then evaluated and assigned a different priority. In later rounds of the judgment process, documents are then selected by considering the run quality first and then their positions in the ranked lists. Although the "pool-to-d" method adopted by TREC ignores the difference between topics, it is an important factor affecting both judgment cost and quality. Cormack et al. also considered applying ISJ on a per-topic basis, in which case the judgment budget has been made for a set of topics instead of individually. Although the ISJ method may work well, it is difficult to apply in practice, and one reason is that the process requires a human assessor to stay in the loop. Also, there is a requirement on the estimation of the quality of a ranked list, meaning that the choice of evaluation metric is also a key issue.

Targeted Relevance Judgments. A third method also considers the probability of different documents being relevant to the topic. The targeted relevance judgment method aims at reducing residuals against certain optimization criteria. Moffat et al. [75] proposed three methods to select documents based on the evaluation metric *Rank-Biased Precision* (RBP) [74]. Considering the n ranked lists submitted by contributing systems in Figure 2.6, it is clear that a document may appear multiple times across n lists at different ranks. While both top-d pooling and ISJ ignore these duplicates, it is a feature suggesting the importance of a document. The first method proposed by Moffat et al. [75] makes use of this information and considers the total contribution of a document by summing its contributions at all rank positions. Since all ranked lists are sorted, different weightings are given to ranks indicating the importance. Let $W_{\mathrm{M}}(i)$ be the weighting function of rank i and let $\mathsf{k}_{D,i}(1 \leq i \leq n)$ be the rank of document D given by the i-th contributing system. Then the contribution of D is calculated as:

$$\mathit{Contrib}(D) = \sum_{i=1}^{n} W_{\mathrm{M}}(k_{D,i}).$$

The Method A proposed by Moffat et al. selects the $|J|$ documents with the highest contribution, where $|J|$ is the judgment budget. This method was designed to optimize toward finding the most popular documents in the candidate set to be judged since they have a high likelihood of being relevant. The Method B proposed by Moffat et al. leverages the information of unjudged documents in a system instead of only relying on the set of judged doucuments. Let J be the set of judged documents and $s_{j,i}$ be a document returned by system i at position j, then Method B

scores the contribution of a document as:

$$\textit{Contrib}(D) = \sum_{i=1}^{n} W_{\mathrm{M}}(k_{D,i}) \cdot \Delta_i, \text{ where}$$
$$\Delta_i = \sum_{\substack{j=1 \\ s_{j,i} \notin J}}^{\infty} W_{\mathrm{M}}(j)$$

where Δ_i accounts for the total weightings of unjudged documents in system i. By doing so, documents are selected not only to maximize the amount of relevant documents, but also to minimize the uncertainties in system evaluations. A third method proposed by Moffat et al. requires some knowledge of the relevance of selected documents, which is similar to the idea of Cormack et al. [28]. The major difference between Method C [75] and ISJ [28] is that it estimates an overall system quality:

$$\begin{aligned} \textit{Contrib}(D) &= \sum_{i=1}^{n} W_{\mathrm{M}}(k_{D,i}) \cdot \Delta_i \cdot (\textit{Base}_i + \Delta_i)^{3/2}, \\ \textit{Base}_i &= \sum_{\substack{j=1 \\ s_{j,i} \in J}}^{\infty} r_j \cdot W_{\mathrm{M}}(j). \end{aligned} \tag{2.14}$$

where r_j is the relevance of document at rank j. Obviously, the additional component in Method C estimates the system quality and thus the last method assigns the weightings to a document using all three aspects. A recent study conducted by Lipani et al. [58] compared the three collection construction methods and also the pooling-to-d method. Experimental results show that Method A and Method C perform the best in constructing reusable test collections, however, Method C is difficult to implement in a one-pass mechanism, but is possible with several judgment rounds.

Summary. Table 2.3 summarizes the widely discussed methods of selecting documents to be judged. A top-d pooling method is required in most sampling-based approaches, for example the first stratum is also a shallow top-d pool in the stratified sampling process, so does the approach proposed by Yilmaz et al.. Column "Prioritizing" indicates whether the corresponding approach assigns document weightings according to their contributions, when considering ranked lists from all contributing systems. Both ISJ [28] and methods in Moffat et al. [75] are based on this since all documents are prioritized according to some weighting schemes. Also, ISJ and Method C require human assessors to continue judging the documents during the document selection, which is indicated as "Assessors". While most of the document selection process is independent of the evaluation process, methods proposed by Yilmaz and Aslam [127], Yilmaz et al. [130], Aslam et al. [5] and Carterette et al. [20] are not, which are marked as "Metric Dep.". The resulting test collections constructed using these methods are required to be evaluated with specific metrics.

2.4.2 Relevance Values

Binary and Graded Relevance. After the set of documents J has been chosen, they will be shown to an assessor, often the same person who created the topic. For each document, a relevance

Method	Sampling			Prioritizing	Metric Dep.	Assessors
	Uniform	Stratified	Other			
Top-d [118]						
ISJ [28]				✓		✓
Inf. Metrics [127]	✓				✓	
Extended Inf. Metrics [130]		✓			✓	
statAP [5]			✓		✓	
MTC [20]			✓		✓	
Method A [75]				✓		
Method B [75]				✓		
Method C [75]				✓		✓

Table 2.3: Summary of different judgment set construction methods. the "Sampling" column categorize the type of sampling method used in each collection construction method; the "Prioritizing" column indicates whether documents are prioritized by the contributions made; the "Metric Dep" column marks whether a specific evaluation metric is required when using the judgments; the last column "Assessors" marks method requires human-in-the-loop.

value will be given by the assessors. These relevance values will be subsequently used in order to calculate effectiveness scores in the evaluation stage. In early TREC tasks, a binary decision was usually made by assessors indicating a document as relevant or not. However, the process of assigning relevance is subjective [52], and using binary relevance may result in unexpected issues. Kekäläinen and Järvelin [52] argued that the most significant drawback is that it can hardly distinguish systems returning highly relevant documents and their experimental studies conducted on multiple information tasks suggested that using graded relevance in IR evaluations was preferable in many cases. Typically graded judgments use a set of labels such as {not relevant, marginally relevant, relevant, highly relevant} or in the corresponding numeric values {0,1,2,3}.

Gain Mapping Functions. The benefit of using graded relevance in building test collections also requires the development of evaluation metrics which accept graded relevance in the calculation. In order to make use of graded relevance, a gain mapping function that converts a relevance value assigned in the judging process to a numeric gain score in the evaluation process is required. Let *max_rel* be the predefined label indicating the maximal relevance level, *min_rel* be the minimal relevance level and *rel* be a relevance value assigned to the document, ranging from *min_rel* to *max_rel*. A mapping function that computes relevance values to be the probability of the document being relevant [23] is required in computing effectiveness metrics and there are at least three commonly used gain mapping functions. The first is a *binary* mapping function:

$$gain_b(rel) = \begin{cases} 1, & \text{if } rel \geq \theta \\ 0, & \text{otherwise} \end{cases}, \tag{2.15}$$

where a relevance value *rel* is accepted as an input and a binary decision is output, relevant to a predefined threshold θ. Although binary gains are widely used in official evaluation tools such as

trec_eval[8], their use is not recommended unless necessary, since it may be difficult for researchers to conclude whether a system is better than another at a particular degree of relevance [52]. As defined in Equation 2.15, in the binary gain mapping process, a predefined threshold is required, and $\theta = 1$ is often employed. Also, by using a binary gain mapping, the benefit of graded relevance no longer exists. A second gain mapping option is a linear mapping:

$$gain_l(rel) = \frac{rel}{max\_rel}\,, \tag{2.16}$$

The linear gain mode is readily applied in most metrics and gives equal weight difference between each pair of adjacent levels of relevance values. If a more distinguishable gap is desired between a highly relevant and a marginal value, an exponential gain mapping function may be used [23, 48, 51], which is computed as:

$$gain_e(rel) = \frac{2^{rel} - 1}{2^{max\_rel}}\,, \tag{2.17}$$

2.4.3 Evaluation Metrics over Complete Judgments

Classic Ad Hoc evaluation components are shown in Figure 2.7. Each small rounded rectangle in Figure 2.7 indicates an evaluation metric and different connecting lines indicate relationships among them. The dashed lines indicate a "based on" relationship, which means two metrics have connections with each other. An "Extended" relationship means that an approach is extended from which the arrow points to. A solid line, which is "use", marks a relationship where one building block uses another in order to accomplish a subgoal. Finally, a dashed or a solid arrow shows the conversion between two methods is achieved by using different sets of judgments. The central dashed rectangle in Figure 2.7 shows commonly used core Ad Hoc evaluation metrics.

Recall and Precision. Amongst all Ad Hoc evaluation metrics, recall (Recall) and precision (Prec) are the two most basic ones. Prec measures the number of relevant documents retrieved relative to the total number of returned documents in a ranked list. However, the rank information will be ignored during the computation, regarding the ranked list as a set:

$$\text{Prec@}L = \frac{\sum_{i=1}^{L} gain_b(r_i)}{L}, \tag{2.18}$$

where the binary gain mode is applied and L is the total number of documents returned by a retrieval system. If only a few top ranked documents are taken into consideration, then we substitute L to the maximal considered rank position k. Recall measures the fraction of retrieved relevant documents relative to the total number of relevant documents of this topic:

$$\text{Recall@}L = \frac{\sum_{i=1}^{L} gain_b(r_i)}{R}, \tag{2.19}$$

where R is the total number of relevant documents in the document collection for this topic. When only precision and recall are considered as evaluation metrics, both of them are reported in

[8] http://trec.nist.gov/trec_eval/index.html

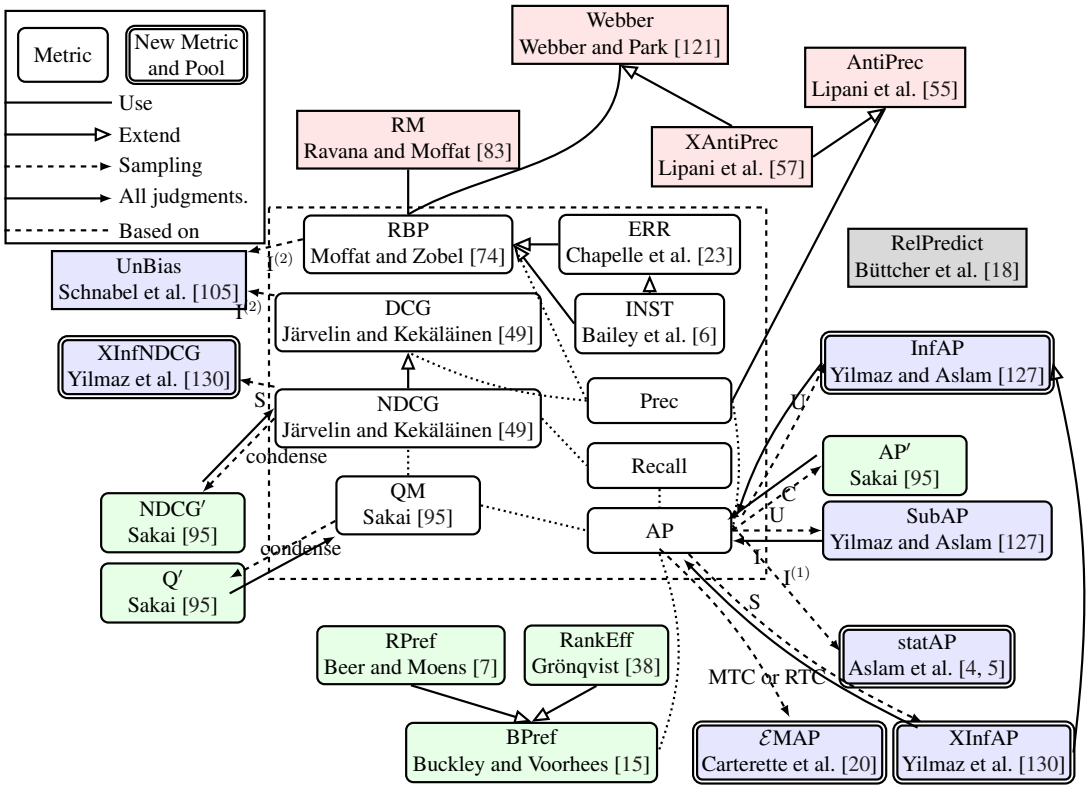


Figure 2.7: An overview of evaluation metrics and approaches. The metrics located in the central pane are classic Ad Hoc evaluation metrics. All the colored rectangles indicate methods to deal with the incompleteness of judgments. Rectangles in the same colors represent similar approaches. The double borders indicate the methods require an additional judgment process other than a pooling method. The "I" and "U" plotted on the dashed arrow indicate the importance sampling and uniform sampling, respectively. A "use" relation means current methods "use" metric linked to. An "Extend" relation means one method is derived from another. The "All judgments" relation indicates that the method is equivalent to another when all judgments are available.

the form of P-R curves in order to conclude performance of a system. Using Recall and Prec for evaluating ranked retrieval ignores the ranking positions, since they were originally designed in the context of Boolean retrieval evaluation. In particular, when two systems return the same number of relevant documents, there is no difference in effectiveness when evaluating with the two metrics. As ranked retrieval becomes more popular, making use of the ranking information becomes more of the focus in developing new evaluation metrics for ranked retrieval. Based on the intuition that the system which puts the relevant documents into more highly ranked positions should be considered as a good retrieval system, more evaluation metrics developed for ranked retrieval have been proposed, all of which have Prec or Recall or both properties, as shown in Figure 2.7. Whether or not their computations contain recall components gives rise to two key categories: (i) weighted-precision metrics, which are only based on precision metrics; and (ii) recall-based metrics, which are often based on both. Both recall-based and weighted-precision metrics are in common use. The remaining paragraphs describe a range of variants in each category.

Metric	Discount Mode	*gain*(*rel*)	$W_{\mathrm{M}}(i)$
Prec	N/A	binary	$1/L$
DCG [48]	Log	any	$1/\log_b(b-1+i)$
	Linear	any	$L+1-i/L$
	Zipf	any	$1/i$
RBP [74]	N/A	any	$(1-p)\cdot p^{i-1}$
ERR [23]	N/A	any	$\frac{1}{i}\cdot\prod_{j=1}^{i-1}(1-\mathit{gain}(r_j))$

Table 2.4: Weighted-precision metrics and the calculation components.

Weighted-Precision Metrics. Let $\mathrm{M}(S)$ be a weighted-precision evaluation metric, where S is a retrieval system which returns L ranked documents. In general, their calculation can be written in the form:

$$\mathrm{M}(S) = \sum_{i=1}^{L} W_{\mathrm{M}}(i)\cdot \mathit{gain}(r_i), \tag{2.20}$$

where r_i is the relevance value of the document at rank i, and *gain*(r_i) is a gain mapping function. Table 2.4 shows several widely applied weighted precision metrics. As shown in Table 2.4, all weighted-precision metrics except Prec accept all gain modes. The $W_{\mathrm{M}}(i)$ component in a weighted-precision metric usually represents a user model [74]. It is a function of retrieval depth that discounts the gain values at the current considered rank position.

Although Prec is an important evaluation metric, it does not leverage the graded relevance and is not the best option for evaluating ranked retrieval systems. Considering both ranked retrieval and graded relevance settings, Järvelin and Kekäläinen [48] proposed Discounted Cumulative Gain (DCG) based on two hypotheses:

1. Highly relevant documents are more valuable than marginally relevant documents.
2. The deeper the ranked position of a relevant document, the less valuable it is for the user.

The original calculation of DCG uses an exponential mapping function $\mathit{gain}_e(\mathit{rel})$ and a $\log$-type discount function $W_{\mathrm{M}}(i)$:

$$\begin{aligned} \mathrm{DCG} &= \sum_{i=1}^{L} \frac{\mathit{gain}_e(r_i)}{W_{\mathrm{M}}(i)}, \text{ where} \\ W_{\mathrm{M}}(i) &= \begin{cases} i, & \text{if } i < b \\ \log_b(i), & \text{otherwise} \end{cases} \end{aligned} \tag{2.21}$$

The discount function reduces the document score as its rank increases, showing some level of user persistence [49], and the logarithmic function is the simplest and most straightforward among all

options. Moreover, the $\log$-type discount function itself was designed with the expectation to be able to model different users by varying the $\log$ base. A large base value implies that a user will at least examine the ranked list until the base rank is reached. In DCG, there is no assumption on how a user will examine a ranked list, while Moffat and Zobel [74] models user's patience via a parameter p. A large p-value describes the behavior of a patient user who has a higher probability of examining items occurring at greater depths within the ranked list, while a smaller value of the parameter p models an impatient user, who has a higher probability of ceasing to examine any further documents other than those within the top few ranked positions. System performance should thus be evaluated based on the two behaviors. In order to do so, Moffat and Zobel [74] proposed Ranked-biased Precision (RBP), with a persistence parameter p indicating the user behavior type:

$$\text{RBP} = (1-p)\sum_{i=1}^{L} p^{i-1} \cdot \mathit{gain}_*(r_i), \tag{2.22}$$

in which any of the modes listed in Table 2.4 can be applied. As indicated in the calculation, the probability that a user stops at any rank is independent of any relevance value, but follows a predetermined geometric distribution parameterized by p. The important feature that distinguishes RBP from previous metrics is its ability of quantifying uncertainties in the evaluation process, by using a *residual*:

$$\Delta = (1-p)\sum_{\substack{i=1 \\ S_i \notin J}}^{\infty} p^{i-1} \cdot \mathit{gain}_*(r_i), \tag{2.23}$$

in which S_i is the document returned by a system S at position i and J is the set of judgments. Note that the $\mathit{gain}_b(r_i)$ should be matched with the one used in Equation 2.22. As indicated by the calculation, a residual is the maximum possible effectiveness score a system can achieve if all unjudged documents are relevant. Hence, RBP is not merely a metric providing a single numerical value, but a range of possible values that quantify system effectiveness. This feature makes it possible, for example, to guide a strategic judgment building process, which was proposed by Moffat et al. [75]. If a top-d pooling scheme is to be employed, the residual can also be a guide for choosing the pooling depth within a predefined error range. For example, if RBP with p (RBP(p)) is applied in the evaluation, then, in order for the residual value to be less than 10^{-4}, the pooling depth d should be at least [74]:

$$d \approx \frac{9.21}{1-p}.$$

A third popular weighted-precision metric is Expected Reciprocal Rank (ERR) [23]. The main difference between ERR and RBP in terms of modeling users is the assumption as how a user stops examining documents within the ranked list. However, the assumption that the probability that a user examines the next document within the ranked list is decreased at ranks where relevant documents appear, entails the worst-case of the metric to be the computation of an infinite ranked list. While RBP makes probability that a user stops examining a ranked list independent of whether a relevant document will be examined, ERR assumes the probability of whether a user continues

examining the ranked list will decrease if relevant documents have been examined:

$$\text{ERR} = \sum_{i=1}^{L} \frac{gain_e(r_i)}{i} \cdot \prod_{j=1}^{i-1} (1 - gain_e(r_j)), \tag{2.24}$$

where the gain mapping function $gain_e(r_i)$ is used as an illustration since it is originally defined as a computational component, but as listed in Table 2.4, any other gain mapping functions are applicable. Note that, although a binary mapping function can be applied to ERR, it is not recommend since using a binary relevance value is equivalent to Reciprocal Rank – the reciprocal value of the rank position where the first relevant document occurs. One major property of all weighted-precision metrics is their non-decreasing property, adding on judgments will not make the score decrease. This implies that the effectiveness score obtained by weighted-precision metrics can be regarded as a lower bound of system performance, when limited judgments are available. It also suggests that a residual can be computed for all weighted-precision metrics, which is not restricted to RBP alone.

Recall-Based Metrics. As discussed, recall-based metrics contains a recall component, either implicitly or explicitly. By definition, the recall component requires the knowledge of the total number of relevant documents, which is often a challenge if a test collection is not exhaustively judged. One of the most widely applied metrics is Average Precision (AP), which can be viewed as the expected precision at ranks where a relevant document appears. It is calculated using binary gains as:

$$\text{AP} = \frac{\sum_{i=1}^{k} gain_b(r_i) \cdot \text{Prec@}i}{R}, \tag{2.25}$$

where R is the total number of relevant documents and Prec@i is the precision at position i. Moreover, AP makes use of the binary relevance gain function in the original definition, but later, Robertson et al. [91] extended it to graded relevance. Robertson [86] proposed probabilistic interpretation for AP, which assumes the stopping ranks of the user is uniformly distributed over all ranks at which the R relevant documents are located. This user model can also be connected to the probabilistic interpretation provided by Yilmaz and Aslam [127] and is differentiated from the one used in RBP [74], where the stopping probability of a user is pre-determined by setting a persistence value. So far, we consider the calculation per topic. As the typical IR batch evaluation setting is to use a set of topics, therefore, an average AP score is often taken over the set of topics. Although the Mean AP (MAP) is often calculated in the evaluation process, Robertson [85] also suggested other alternative aggregation techniques. Among all those alternatives, Geometric Mean AP (GMAP) is the combination that has sometimes been applied in practice. Computing the geometric mean rather than the arithmetic mean attenuate the evaluation more toward topical difference, and may give a clearer guide on improving systems.

As the original design of AP can only be used with binary relevance and given that graded relevance judgments provide more discrimination power between systems, new recall-metrics that can be applied with graded relevance are also desirable. Motivated by this, Järvelin and Kekäläinen [49] proposed Normalized DCG (NDCG), which represents the recall component implicitly by

Rank:	1	2	3	4	5	6	7	8	9	10	...	20	...	100	...	200	...	1000
S_1	1	1	1	0	0	0	0	1	0	1	...	1	...	1	...	0	...	0
S_2	0	0	1	0	0	1	1	1	0	1	...	1	...	0	...	0	...	0
S_3	1	0	0	1	0	1	1	0	1	1	...	0	...	1	...	1	...	1

Table 2.5: Three example relevance vectors. Assume the pooling depth $d = 100$ and $R = 20$.

including a normalization step determined by the DCG score of an ideal ranked list. The ideal ranked list is defined as all documents in the document collection sorted in decreasing order of relevance values. Therefore, this actually can be viewed as the "total available relevance gain" for a topic. Let r'_i be the relevance value of the document at position i of an ideal ranked list, then NDCG is calculated as:

$$\text{NDCG} = \frac{\sum_{i=1}^{L} \mathit{gain}_e(r_i)/\log_b(b+i-1)}{\sum_{i=1}^{L} \mathit{gain}_e(r'_i)/\log_b(b+i-1)}, \tag{2.26}$$

where the log-type discount function can be replaced by linear or Zipfian options listed in Table 2.4 if required. Also, the gain mapping function is often the exponential gain mapping, but other alternatives can also be applied. Note that although the denominator seems similar to the calculation form of the numerator, it is based on the ideal ranked list instead of the one actually returned by the system being measured.

Sakai [93] proposed a new metric based on graded relevance – QM. Sakai argued QM is more reliable than DCG and can be used for question answering (QA) tasks. QM has the characteristics of both NDCG and AP and blends the two using a parameter β:

$$\text{QM} = \frac{1}{R}\sum_{i=1}^{L} \mathit{gain}_b(r_i) \cdot \frac{\sum_{j=1}^{i} \mathit{gain}_b(r_j) + \beta \sum_{j=1}^{i} \mathit{gain}_l(r_i)}{i + \beta \sum_{j=1}^{i} \mathit{gain}_l(r'_j)}, \tag{2.27}$$

where r'_j is the relevance of a document at position j of an ideal ranked list. There are two different gain modes used in QM, binary and linear, which corresponds to the component from AP and NDCG, respectively. When $\beta \to 0$, QM becomes AP and when $\beta \to 1$, it behaves like NDCG. Although Sakai argued that QM was designed for QA tasks, it is also used in more general Ad Hoc evaluations.

Truncation in Evaluation Metrics. All evaluation metrics are normally defined and calculated over an entire ranked list submitted by a retrieval system. But the more usual case is for them to be applied to top-k ranked documents. In this case, all of the above metrics are expected to only work with a prefix of the ranked list rather than in its entirety. Consider an example of three ranked lists in Table 2.5. Suppose that, there are 20 relevant documents in the judgment set and the pooling depth is $d = 100$. Let the evaluation depth be k, which means only the prefix-k relevance values will be taken into consideration when computing the system effectiveness scores. Consider Prec first, the calculation of Prec@k:

$$\text{Prec@}k = \frac{\sum_{i=1}^{k} \mathit{gain}_b(r_i)}{k}. \tag{2.28}$$

If $k = 10$ is used in our running example, then S_1 and S_2 will have the same effectiveness score, which is $5/10$, while S_3 will have a score of $6/10$. However, if an evaluation depth $k = 3$ is applied, then S_1 will be the best with a score of 1. It is obvious that a different k makes Prec a different metric, and gives different results in system evaluations. It is also easy to evaluate system effectiveness in truncated lists using weighted-precision metrics, because the scores are strictly additive. Therefore, the general calculation form of a weighted-precision metric:

$$\text{M@}k = \sum_{i=1}^{k} W_{\text{M}}(i) \cdot \mathit{gain}_*(r_i), \tag{2.29}$$

where both $W_{\text{M}}(i)$ and $\mathit{gain}_*(i)$ are the same as listed in Table 2.4. Take DCG as an example, when only $k = 3$ is considered, then S_1 is scored as 3.13 and S_2 is scored as 0.5. When a $k = 10$ is used, then S_2 has a score of 7.81 and S_1 has score of 8.76. It is clear that when a weighted-precision metric is applied in evaluation, if and only if there are more relevant documents appearing in the evaluation depths under consideration, the effectiveness score of a system will be affected. Also, due to the discount factor, documents appearing at very deep ranks may only have a small impact on the evaluation results.

However, it is less clear how Recall and recall-based metrics should be computed when only top-k relevance values are considered. For example, we can simply truncate Recall by using:

$$\text{Recall@}k = \frac{\sum_{i=1}^{k} \mathit{gain}_b(r_i)}{R}, \tag{2.30}$$

which is the fraction of retrieved relevant documents so far relative to the total number of relevant documents in the document collection. The definition of Recall@k has been widely applied in practice but it is less clear that how other recall-based metrics such as AP or NDCG should be evaluated under a prefix-k configuration.

There are three widely applied truncated AP variants. The first one is defined in a similar way to weighted-precision metric truncation:

$$\text{AP}_a\text{@}k = \frac{\sum_{i=1}^{k} \mathit{gain}_b(r_i) \cdot \text{Prec@}k}{R}, \tag{2.31}$$

where R is the total number of relevant documents in the collection. However, one drawback of using this method is that a "perfect" system at current evaluation depths can never receive a perfect score of 1. For example, in the example in Table 2.5, when $k = 3$, although S_1 returns three relevant documents in top ranks, its effectiveness score does not reflect this case. While TREC evaluation adopts the calculation in Equation 2.31, the evaluation tool provided by NTCIR uses an alternative configuration:

$$\text{AP}_b\text{@}k = \frac{\sum_{i=1}^{k} \mathit{gain}_b(r_i) \cdot \text{Prec@}k}{\min\{R, k\}}. \tag{2.32}$$

Consider the same configuration and example as above, S_1 can now obtain a score of 1.0, which is because the normalization factor has been changed to $k = 3$ in the calculation. Both variations

assumes that a global R can be obtained or approximated. However, this assumption has been challenged and so a third option has arisen:

$$\text{AP}_c@k = \frac{\sum_{i=1}^{k} \mathit{gain}_b(r_i) \cdot \text{Prec}@k}{R_k}, \tag{2.33}$$

where R_k is used to denote the value of R that emerges after pooling is carried out to depth k, so that $R_0 = 0$, $R_k \leq R_{k+1}$ and $\lim_{k\to\infty} R_k = R$.

There is no consensus on defining truncated AP computations. Also, when only considering an "evaluation-to-k" configuration, how to calculate recall-based metrics is an important issue. The same situation also applies when defining a truncated version of NDCG. An obvious version is the same as $\text{AP}_a@k$, which is:

$$\text{NDCG}_a@k = \frac{\sum_{i=1}^{k} W_{\text{M}}(i) \cdot \mathit{gain}_e(r_i)}{\sum_{i=1}^{k} W_{\text{M}}(i) \cdot \mathit{gain}_e(r'_i)}, \tag{2.34}$$

where r'_i is the relevance value at an ideal ranked list position i. Although the idea of defining the NDCG truncation is similar to $\text{AP}_a@k$, the behavior is actually closer to $\text{AP}_b@k$. Consider S_1 and $k = 3$ again, using this calculation suggests the system is "perfect". An alternative, $\text{NDCG}_b@k$, can be defined with a less "ideal" ranked list, where only documents from the top-k ranks are considered. The truncated version of QM has the same issues found in both AP and NDCG truncations. However, the official implementation provides clear guidance on the computation:

$$\text{QM}@k = \frac{1}{\min\{R, k\}} \sum_{i=1}^{k} \mathit{gain}_b(r_i) \cdot \frac{\sum_{j=1}^{i} \mathit{gain}_b(r_j) + \beta \sum_{j=1}^{i} \mathit{gain}_l(r_i)}{i + \beta \sum_{j=1}^{i} \mathit{gain}_l(r'_j)}. \tag{2.35}$$

Summary. All metrics discussed so far make the Cranfield assumption, that is, all documents are labeled with relevance. While it is not true in practice, Voorhees [115] argued that pooling is a good approximation of the total relevant documents, thus recall-based metrics can give a relatively reliable system evaluation. However, the truncated evaluation methods should be applied with caution, especially when using recall-based metrics, since multiple interpretations on how to compute the scores exist.

2.4.4 Evaluation Methods over Incomplete Judgments

As argued by Voorhees [115], a good approximation can be achieved when using a pooling method to build a test collection, and this was confirmed by Zobel [134]. On deeply pooled collections such as NewsWire, there are less bias when evaluating new systems since the majority of relevant documents have been found in the pooling process. However, due to the increasing size of document collections, forming a deep judgment pool is more expensive than on previous NewsWire collections. Therefore, whether the set of judgments can be regarded as an approximation of the complete judgments is questionable. Buckley et al. [16] showed the bias in using incomplete judgments during evaluation. Later, two types of bias were identified in system evaluation when using a set of incomplete judgments [96, 97]. The first is pooling depth bias, which exists especially in an extended evaluation process; the second is system bias, which is the bias occurring between contributing and non-contributing systems.

Solutions to Unbiased Evaluations. We first consider the three simplest methods applied on a set of incomplete judgments: (i) treating all unjudged documents as not relevant; (ii) using a residual to quantify the maximum unknown effectiveness; and (iii) removal of unjudged documents. A default option for handling unjudged documents is to regard them as not relevant, and a system gains nothing when returning unjudged documents. We consider the example systems listed in Table 2.5. Suppose all documents ranked at positions $10 < k < 20$ are unjudged for systems S_2 and S_3, and we use a simple evaluation metric Prec@20. Although both systems have the same score, there is no guarantee that the two systems return the same number of relevant documents, due to the system variability. Therefore, a second option is to use a residual quantifying the level of uncertainty in the evaluation. For example, although S_2 scored less than S_3 based on current "known" judgments, the residual values indicate that the score of S_2 could be benefit from the unjudged documents. The third approach is to apply any metrics on a *condensed* run; that is deriving runs from which all unjudged documents have been omitted. Because this method can be applied to any metric, it is denoted using a superscript $'$ in Figure 2.7. Note that the Induced AP (IndAP) proposed by Yilmaz and Aslam [127] is indeed the AP$'$.

Binary Preference Metric and Variants. The metric, Binary Preference (BPref), proposed by Buckley and Voorhees [15] is the one that was explicitly developed for use with condensed runs. If a query has R_d known relevant documents, then BPref locates the first R_d judged-nonrelevant documents too, and then pairwise compares their ranks against the ranks of the R_d judged-relevant documents. Let S be a ranked list returned by a system and s_i be a document at rank position i, BPref is then calculated as:

$$\text{BPref} = \frac{1}{R} \cdot \sum_{\substack{i=1 \\ s_i \in R}}^{L} 1 - \frac{\min\{|\bigcup_{j<i}\{s_j \mid \mathit{gain}_b(r_j) = 0\}|, R\}}{R}, \tag{2.36}$$

where J is the judged set of documents and R is the number of unjudged documents. Buckley and Voorhees examine the metric is robustness with regard to incomplete judgment by reducing the pool size using a uniform sampling approach. If the set of judgments are nearly complete, then BPref is highly correlated with AP.

However, the calculation of BPref has two weakness: (i) only R judged non-relevant documents are considered, and (ii) only binary relevance is used in the evaluation process. Regarding the first problem, in the original paper, an enhanced version of BPref considered ten more non-relevant documents [15], has been proposed. However, it still loses information from the non-relevant documents to some extent. In order to compensate this, Grönqvist [38] proposed metric *Ranking Efficiency* (RankEff), which is:

$$\text{RankEff} = \frac{\sum_{i=1, r_i>0}^{L} |\bigcup_{j<i}\{s_j \mid \mathit{gain}_b(r_j) = 0\}|}{R \cdot (|J| - R)}. \tag{2.37}$$

Compared to BPref, RankEff consider all judged documents by using all judged non-relevant documents, instead of these from top-R. On a set of topics where the number of known relevant documents is far less than the amount of judged non-relevant ones, BPref loses its ability to

measure system performance while RankEff can still differentiate between two systems. When a similar number of relevant and non-relevant documents have been identified, the two metrics give similar results in system comparisons.

Motivated by Järvelin and Kekäläinen [49], Beer and Moens [7] proposed RPref, which generalizes BPref to graded relevance. The reason is that using graded relevance values, the distance calculation must consider document pairs between which the relevance values differ:

$$\text{RPref} = \frac{1}{\sum_{D \in J} r_D} \cdot \sum_{\substack{i=1 \\ s_i \in J}}^{L} r_i \cdot \left(1 - \frac{\sum_{j=1, r_j < r_i}^{i} (i-j)/r_i}{\sum_{D \in J} 1 - r_D} \right), \tag{2.38}$$

where r_D is the relevance value of document D. Similar to RankEff, RPref also considers all judged documents instead of only a subset. Based on the theoretical analysis, RPref behaves similar to BPref, and is expected to be robust to evaluations conducted on an incomplete set of judgments. However, as is pointed out by Beer and Moens, both BPref and RPref are no more than only a relative ordering of assessed documents, therefore, providing results from another metric is necessary for interpreting system effectiveness.

Inferred Metrics. If a top-d pooling method is not fixed for test collection construction, then a set of metrics using sampled judgments can be applied. It is easier to apply some statistical method to estimate the performance of a system on the sampled judgments than the top-d judgments, because the set of judged documents are selected randomly. As concluded by Hawking and Robertson [41], AP is more stable on a set of randomly sampled subset of the entire collection. Based on this conclusion, Yilmaz and Aslam [127, 128] proposed the Subcollection AP (SubAP). The judgments are formed based on a top-d pooling process used in the TREC framework, with $d \geq 100$. Based on this obtained set of judgments, a re-sampling process has been conducted, and in this process, unjudged documents are considered not relevant. The re-sampling process is performed on both judged and unjudged documents, with a sampling rate of $p\%$. Finally, this re-sampled set of judgments are regarded as J. By definition, the average precision can be interpreted as the expected precision averaged over total number of relevant documents [86]. The known probability $p\%$ makes it possible to calculate the expected precision at rank k. Yilmaz and Aslam compared SubAP to AP′ and BPref on TREC-8 dataset and showed that AP′ achieves a better approximation of AP score calculated using a complete set of judgments.

When considering AP score calculated on the complete judgment set as a gold standard, and in order to approximate this gold standard, Yilmaz and Aslam [127] proposed the inferred AP (InfAP) metric. Similar to SubAP, the core idea behind defining InfAP is to interpret AP as the expected precision over all ranks where relevant documents appear:

$$E(\text{Prec@}k) = \frac{1}{k} \cdot \left(1 + \frac{k-1}{k} E(\text{Prec@}(k-1)) \right). \tag{2.39}$$

Here, depth k is an arbitrary retrieval rank at which relevant documents occur, and $E(\text{Prec@}(k-1))$ represents the expected precision above depth k, which is the key in deriving InfAP. For any unjudged document, there are two different states: (i) the minimum rank position the document

occurs at is less than 100, but the document has not been sampled; or (ii) none of the contributing systems return that document within the top-100. If a document is in the second state, then it will be assumed to be not relevant, while if the document is not sampled, then its relevance will be estimated. The judgments applied in experiments are selected in the same way as Buckley and Voorhees [15]. When only 5% of the original judgments are used, InfAP is still able to achieve a better approximation of AP when compared to BPref. The proposed InfAP has two points of weakness: (i) it provides no confidence intervals; and (ii) the uniform sampling approach is not an ideal option. A follow up study by Yilmaz et al. [130] first devised the confidence intervals for InfAP and then they extended the metric to use stratified sampling judgments. The new metric that incorporates stratified sampling is *eXtended* InfAP (XInfAP). It works by computing InfAP within each stratum and is then aggregated over all strata. The *inferred* NDCG (InfNDCG) [130] is defined in a similar way to InfAP. The key is to reformulate NDCG into standard form of the "expected" DCG value. The ideal gain in NDCG is obtained by estimating the total number of documents at each relevance level. Experimental results on TREC-8/9/10 datasets show the advantage of using a stratified sampling approach. However, one obvious concern is that the sampling configuration may affect the metric performance and Voorhees [117] examined this impact. Voorhees mainly compared the estimation quality on the recall component, and concluded that two strata sampling – an exhaustively judged small initial stratum and using a moderate sampling rate to a moderate depth as the second stratum, is a practically effective sampling approach for inferred metrics.

Unlike the uniform or stratified sampling approach that is easy to apply, another family of inferred metrics aims to minimize estimation variance, by considering an optimal sampling distribution. Also based on AP and by reformulating the calculation in an expected form, Aslam et al. [5] proposed *statistical AP* (statAP). The main difference is that statAP considers AP in the following representation:

$$SP = \sum_{i=1}^{k} gain_b(r_i) \cdot \text{Prec@}i = \sum_{i=1}^{k} gain_b(r_i) \cdot \sum_{j=1}^{i} \frac{gain_b(r_j)}{i}, \tag{2.40}$$

where r_i and r_j indicate two random relevance variables, and the equation suggest that samples should be drawn from a distribution based on the relevance values of document pairs. In order to use the sampled judgments to compare multiple systems, Aslam et al. used a scaling factor, which is calculated as the ratio between the required and the empirical sampling distribution.

Although Aslam et al. [5] focused on estimating system performance based on AP, Recall and Prec can also be estimated using the same sampling technique. The experimental results show that, with fewer judgments, statAP can provide an accurate estimation on system effectiveness scores on TREC-7/8/10. Evaluating new systems using the sampled judgments is relatively fair given that the Root-Mean-Square-Error (RMSE) between the estimated and the true score on new systems is close to the RMSE estimated on contributing systems.

Relevance Prediction and Score Estimation. Another approach of evaluating systems using incomplete judgments is to predict the relevance value of unjudged documents. As Buckley et al.

[16] concluded, on large test collection, the limited pooling depth will result in a biased evaluation, where the non-contributing systems may be underestimated. To address this problem, Büttcher et al. [18] considered the predictions of relevance values of unjudged documents and then used the predictions during evaluation. The prediction problem has been reformulated as a binary classification task for which two approaches have been experimented by Büttcher et al., demonstrating that the predicted values can be applied to give a fair comparison among systems, especially for new retrieval systems that are not part of the pool. Similarly, Carterette and Allan [19] considered to make use of the document similarity in order to improve the evaluation process with fewer judgments. Carterette and Allan [19] use cosine similarity and logistic regression to estimate the relevance probability of a document, which is different to the classification methods adopted by Büttcher et al. [18].

Another line of research is to consider adjusting system scores using a set of incomplete judgments, by estimating the potential effectiveness that is obtained from the set of unjudged documents [55, 56, 57, 83, 121]. Because recall-based metrics are not non-decreasing, it is hard to predict how scores will change when adding more relevant documents. Therefore, studies in this category are all based on weighted-precision metrics, since the additional effectiveness score can be directly added to the existing "known" effectiveness. Using the residual component of RBP, Webber and Park [121] and Ravana and Moffat [83] proposed two different score adjustment methods. Webber and Park mainly consider the problem of evaluating new systems using a set of incomplete judgments, while there is no such specific goal of the approach proposed by Ravana and Moffat [83]. Webber and Park proposed two score adjustment methods. The first method is simple and can be applied to all static test collections, which aims at evaluating new retrieval systems that did not contribute to the pool. The main component, an adjustment factor (or the extra effectiveness score), is estimated using leave-one-out experiments. Each contributing system was left out once from the pool, and after all systems have been left out once, the average score difference becomes the adjustment factor. This method implicitly assumes that the ranked list of the new system is drawn from the set of contributing systems, while this may be true in most cases, there is no guarantee. The second method proposed by Webber and Park [121] makes use of the topical difference, adjusting scores based on a set of selected common topics. Therefore, how to select the set of common topics is crucial in this method and may affect the outcome. Also, Webber and Park set $p = 0.8$ when using RBP to evaluate systems, which only takes relatively a small number of documents into consideration.

Also based on the RBP metric, Ravana and Moffat [83] proposed three score adjustment methods, showing that both types of bias may be partially addressed. The first method is to assume a global percentage λ of gain in the unjudged documents. So $\lambda \cdot \Delta$, where Δ is the residual value when evaluating using the set of incomplete judgments, is the extra effectiveness score that can be obtained from the unjudged documents in the system ranked list. But because λ is collection and system dependent, setting a proper value may be difficult. A second interpolative estimator was proposed based on the assumption that the same percentage of effectiveness can be obtained on the unjudged documents as on the judged ones. Let $\hat{\mathrm{M}}$ be the estimated RBP score, then it can be

calculated as:

$$\begin{aligned} \hat{\mathrm{M}} &= \mathit{Base} + \Delta \cdot \frac{\mathit{Base}}{1-\Delta}, \text{ where,} \\ \mathit{Base} &= \sum_{\substack{i=1 \\ s_i \in J}}^{L} W_{\mathrm{M}}(i) \cdot \mathit{gain}_*(r_i), \end{aligned} \tag{2.41}$$

in which *Base* is the metric score calculated on the set of judged documents using Equation 2.22 and Δ is the residual for that metric. A third method is to smooth between the previous two approaches using a smoothing parameter α:

$$\hat{\mathrm{M}} = \alpha \cdot (\mathit{Base} + \Delta \cdot \frac{\mathit{Base}}{1-\Delta}) + (1-\alpha) \cdot \mathit{Base}, \tag{2.42}$$

where *Base* is calculated in a same way as in Equation 2.41. Their experiments were performed on the TREC-9 test collection, and the reduced set of judgments was obtained by taking $|J'|$ judgments from a complete set. On this dataset, the second method always overestimates system performance when a small set of judgments is used. Although the behavior of the last method is unpredictable, it has a smaller estimation error than Equation 2.41. The experimental results indicate that all three approaches can help with the pooling depth bias to some extent, but there is no clear empirical evidence that the system bias is reduced.

Lipani et al. argued that it is unclear how the first method proposed by Webber and Park [121] can be generalized to Prec. Motivated by this, Lipani et al. [55] proposed "anti-Precision", which is defined as:

$$\text{anti-Prec@}k = \frac{|s_i \mid r_i = 0, 1 \leq i \leq k|}{k}, \tag{2.43}$$

where s_i is the document returned by system S at position i. This counts the non-relevant documents instead of relevant ones, compared to Prec@k. Let Prec and $\overline{\text{Prec}}$ be the effectiveness score of a system measured by precision and anti-precision, respectively, and let the superscript $'$ be scores evaluated using incomplete judgments, then the relationships among three key indicators *diff*, Prec and $\overline{\text{Prec}}$ [55] are:

$$\begin{aligned} &\Delta = -\delta \mathit{diff} = \delta\overline{\text{Prec}}' - \delta\text{Prec}' \\ &\delta\text{Prec}' = \text{Prec}' - \text{Prec},\ \delta\overline{\text{Prec}}' = \overline{\text{Prec}}' - \overline{\text{Prec}}\,. \end{aligned} \tag{2.44}$$

Consider the residual calculation process, then $\delta \mathit{diff}$ defined in the original paper [55] is the Prec residuals, and we denote it as Δ to be consistent with the previous notation. Lipani et al. merged the three indicators into a single value based on empirical observations, and the final value is also used to determine whether an adjustment should be performed. Just as Webber and Park [121], a leave-one-out experiment is used for all systems that contributes to the pool, and an averaged Δ over all pooled systems is computed as the adjustment factor. An extensive study has been conducted on multiple deeply pooled test collections, showing that the proposed method can adjust the system bias and give relatively fair comparison results.

2.4.5 System and Metric Comparisons

System comparisons are performed based on the metric scores computed over the set of judgments. In batched evaluation settings, the evaluation process is performed over a set of topics, and usually a mean metric score is used as evidence for concluding whether or not a system is better than another. In order to make a more reliable conclusion in regard to system differences, statistical tests are often employed which we will discuss in this section. A second evaluation scenario that often attracts the attention of researchers is the meta-evaluation of metrics. When performing metric comparisons, the rank correlation is usually our focus and we measure whether a different conclusion will be made if a different evaluation metric is employed. Metric comparison is another focus in this section.

Significance Testing. The main goal of the evaluation process is to determine whether one system is better than another. While this conclusion can be reached by simply comparing evaluation scores for the metrics discussed in the previous sections, the difference may not be significant and may happen by chance. Therefore, it is important to tell whether the observed difference between the two systems are true or by chance [44]. A widely adopted approach is to conduct statistical testing between two systems across a set of topics. Based on the outcome, a conclusion must be reached as to the generalizability of the observed difference. After a p-value has been calculated in the significance test, a threshold θ ($\theta = 0.05$ is widely used) is set and when $\theta < p$, the two compared systems are noted as being significantly different.

There are several popular choices of significance testing approaches [44]: the Student t-test; the Wilcoxon signed test; the sign test; randomized testing; and bootstrap-based tests [94]. Smucker et al. [107] compare and discuss the problem of choosing a statistical testing methods when comparing systems. On a set of 50 topics, Smucker et al. [107] conduct randomized experiments in order to observe the agreement between two tests on system orderings. The experimental results show that except for the Wilcoxon and signed test, all other significance tests lead to similar conclusions when comparing systems. The conclusion obtained via a Wilcoxon or a signed test was sometimes misleading and not generalizable. Both randomized testing and bootstrap testing provide conclusions beyond the mean effectiveness scores of two systems. Smucker et al. suggest combining a randomized test and a t-test statistic value for more reliable system comparisons. However, when a computed p-value does not achieve the predefined threshold, there is no conclusive result for the comparison.

Statistical Power and Effect Size. One crucial component in statistical testing is whether a desired power of the experiments can be achieved. Webber et al. [122] propose a hybrid approach for determining the number of topics that should be used in order to obtain a target statistical power. Webber et al. also consider the effect size of different pooling depths, and concluded that a shallow judgment pool and a large number of topics is preferable to a deep judgment pool on a small set of topics. Sakai [100] recently examines the statistical power of experiments in published papers. A majority of the research papers reported under-powered experimental configuration and only a few papers published using industrial datasets are over-powered. Sakai [100] further

suggests reporting the effect size in addition to the p-value, since a large effect size can also result in a small p-value. Another key factor in drawing a conclusion in system comparisons is the number of topics used in experiments. Both studies by Webber et al. [122] and Sakai [99] consider the question of the required topic set size in order to have a reliable system comparison, and point out that 50–100 topics may be sufficient.

Metric Stability. Different metrics focus on different perspectives when evaluating system effectiveness and metrics are different in their ability to give a stable orderings of a set of systems. Buckley and Voorhees [14] proposed *error rate*. In order to calculate the error rate, a *fuzziness* value needs to be predefined. If the difference of mean effectiveness scores between two systems exceeds the fuzziness value, they are considered different. Otherwise, the two systems are considered to be equivalently effective. The error rate is similar to statistical testing, which was discussed in the previous section, but it lacks statistical interpretations. The error rate is defined over a set of repeated random experiments. In each experiment, a subset of topics is selected, and the effectiveness score of a system is computed over the selected set of topics. Let M_i and M_j be metric scores of two retrieval systems S_i and S_j and for simplicity, notations "$>$", "$<$" and "$=$" are used to denote "better", "worse" and "equivalently good", respectively. The error rate is computed by repeatedly evaluating the same set of systems over different sets of topics, and it is calculated as the total errors of system pairs divided by total number of possible decisions:

$$\textit{Error Rate} = \frac{\sum_{\langle i,j \rangle} \min\{|\mathrm{M}_i > \mathrm{M}_j|, |\mathrm{M}_i < \mathrm{M}_j|\}}{\sum_{\langle i,j \rangle} |\mathrm{M}_i > \mathrm{M}_j| + |\mathrm{M}_i < \mathrm{M}_j| + |\mathrm{M}_i = \mathrm{M}_j|}, \tag{2.45}$$

where $|\mathrm{M}_i > \mathrm{M}_j|$ is the number of times S_i is better than S_j, and $|\mathrm{M}_i < \mathrm{M}_j|$ is the number of times S_i is worse than S_j, respectively.

Based on the experimental results, Buckley and Voorhees recommend AP to be applied in evaluating system effectiveness as it is stable and discriminating. Also, they point out that it is not sufficient to select an evaluation metric based only on error rate.

As discussed, the error rate is computed with a predefined fuzziness value, which is chosen by considering an absolute difference between two system's effectiveness scores. Since most of the system comparisons are examined using some statistical significance test, it is also worth noticing the relationship between error rate and significance testing. A later work done by Sanderson and Zobel [104] examined this relationship. In the computation, they replace the fuzziness value with significance level p of statistical testing, and establish upper and lower bounds on the error rates.

Discrimination Ratio. The error rate defined in Equation 2.45 can reflect the metric stability to some extent. While giving a stable ranking is an important feature of an evaluation metric, understanding to what extent an evaluation metric can tell the difference between two systems is also important. In order to observe this, the *discrimination ratio* is often calculated and reported [94]. Unlike the fuzziness value that is defined by the absolute difference between two systems, discrimination ratio is calculated based on a predefined significance level p obtained from a statistical test. Let n be the total number of participants, and for simplicity, "$\neq_p$" is used to represent two systems

that are different at significance level p. Then discrimination ratio is calculated as:

$$discr = \frac{|\{S_i \neq_p S_j\}|}{n(n-1)/2}, \ (i \neq j, 1 \leq i, j \leq n) \tag{2.46}$$

which is the fraction of significantly different pairs over all system pairs. Any statistical testing methods can be applied with Equation 2.46. Based on the discrimination ratio, metrics that are only mildly top weighted (or evaluate deep), such as AP, NDCG, and RBP(0.95) tend to have a higher discrimination ratio than shallow metrics such as Prec@5 or RBP(0.5) [98]. This conclusion further confirms the observations made by Buckley and Voorhees. However, it also should be noticed that the deep metrics require more judgment effort comparing to those that are shallower, thus are more expensive. Discrimination ratio should be considered together with features of current test collections, such as the number of topics used in the evaluation, the topical difference.

System Orderings Predictions. A third criterion that might be used to quantify the relative behavior of metrics is their ability to order a set of systems relative to the ordering established by one or more reference metrics. There is no guideline for choosing a reference point for measuring such correlation. Ideally, a system ordering evaluated using the complete set of judgment should be preferred, which is hard to realize in practice. As AP is a dominant metric in system evaluation, many researchers prefer to consider it as the reference point.

When measuring the correlation between two metrics, either system effectiveness scores or system ranking lists or both are considered. Score correlation is often measured using Pearson's correlation coefficient or RMSE. While there are several options used to measure the second type of correlation, Kendall's τ is chosen by most researchers in the IR literature [102]. It indicates the fraction of disagreement on the pairwise system comparison relative to the total number of system pairs. However, Kendall's τ measures nothing beyond how different two system orderings are. An obvious weakness of Pearson's and Kendall's τ is that discordant pairs that occur at the top of the ranking incur the same penalty as the swaps at the bottom of the ranking, whereas in IR, there is often more emphasis on having the correct system ordering at the top of the ranking than there is on having the correct ordering at the bottom. In order to address this weakness, Yilmaz et al. [129] proposed τ_{ap}, which gives more weight to the discordant pairs at top ranked positions. The proposed τ_{ap} weights the swapped system pairs based on AP calculation. Compared to Kendall's τ, if the disagreement appears uniformly across a ranked list, then τ_{ap} is the same as Kendall's τ. Otherwise, τ_{ap} will be larger than Kendall's τ if the disagreement happens at the top ranks and it will be smaller if the error occurs at the bottom rankings. Nevertheless, both correlation measures can only be applied with conjoint ranked lists. In order to measure disjoint lists, Webber et al. [124] proposed Rank-Biased Overlap (RBO). RBO makes use of the same weighting scheme as RBP [74] and is parameterized using p. Similar to τ_{ap}, RBO is top-weighted, which gives a higher weight of system pairs swapped at the top ranks. However, for disjoint and incomplete ranked lists, only RBO is applicable. Other measurements are described in Webber et al. [124].

2.5 Experimental Dataset Description

All of the experiments presented in this thesis have been carried out using TREC test collections. TREC is an annual evaluation conference that not only provides publicly available test collections, but also has had a profound impact on the development of IR techniques. This section details the test collections on various experiments.

```
<DOC>
<DOCNO>FT921-7364</DOCNO>
<PROFILE>_AN-CBVAEADGFT</PROFILE>
<DATE>920222</DATE>
<HEADLINE>
Private View: A mother in waiting to be
a priest-Church leaders debated...
</HEADLINE>
<BYLINE>By CHRISTIAN TYLER</BYLINE>
<TEXT>
DILLY BAKER is expecting to become a
mother in two weeks' time and an
Anglican priest within two years. The
birth of her first child may be ...
</TEXT>
<PUB>The Financial Times</PUB>
<PAGE>
London Page XX Picture (Omitted).
</PAGE>
</DOC>
```

(a) An example TREC document

```
<top>
<num> Number: 301
<title>
International Organized Crime
<desc>
Description:
Identify organizations that participate
in  international criminal activity
...
<narr>
Narrative:
A relevant document must as a minimum
identify the  organization and the
type of illegal activity (e.g.,
Columbian cartel exporting cocaine).
Vague references to international drug
trade  without identification  of
the organization(s) involved would
not be relevant.
</top>
```

(b) An example TREC topic

Figure 2.8: An example TREC document and topic.

2.5.1 Document Collections and Tracks

We briefly listed the TREC tracks in previous sections. In this section, we will further detail the documents and topics used in TREC shared tasks, with an emphasis on the documents and topics used in each track. Also, we present some of the differences between the previous test collections and current test collections used and discuss how problems can arise from such a difference, which is one of the motivations of Chapters 5 and 6.

Document Collections. There are several different document collections distributed by the TREC organizers, spanning news articles, government documents, and web pages. Each document is in an SGML format and is enclosed with <DOC> tags, as shown in Figure 2.8(a). There are several different pairs of tags in a document, each of which indicate a specific field, for example, the <DOCNO> contains a unique document identifier, and the main document content is included by the <TEXT> tags. These document collections were created for specific tasks. Early TREC conferences (TREC-1 to TREC-8) mainly considered documents from news articles and government documents [118], which are relatively small in terms of size. In general, the size of corpus used for each task was around 2GB uncompressed. As web search became the dominate format, developing and evaluating systems for web search tasks attracted the attention of the IR community. Since TREC-9, the web task has been an important track. Although the corpus, wt10g, which is used in TREC-9, is only a subset of documents of VLC2, its size is lerarger than the previously

used news-based test collections. The wt10g corpus contains 1.69 million web pages crawled from web data, aimed at providing and simulating a web search environment while preserving web page characteristics such as hyperlinks [40]. Regarding the scale of a real web search scenario, subsequent web search tasks created new large-scale test collections in order to make it possible for researchers to evaluate retrieval systems on a much larger scale than before. The TeraByte web tracks of 2004–2006 (TB04–06) provided test collections to fulfill these goals, making use of a document collection of 25 million documents (around 426GB uncompressed in size) crawled from the `.gov` domain [25]. The successive web tracks starting from 2009 used a billion page document collection, ClueWeb09, which has a size of 25TB uncompressed [26]. Given the enormous size of the corpus, a smaller subset was also provided for experimental purposes, with the original and subset collections are differentiated with suffix "A" and "B", respectively.

The TREC Ad Hoc Tracks. There are several different tracks in TREC evaluation experiments, and they can be categorized from different aspects based on the search task. The thesis focuses on the Ad Hoc search task, which require retrieval systems return a ranked list of documents obtained from a document collection based on a specific information need of a user. These information needs are termed as *topics* in the TREC framework, as shown in Figure 2.8(b). The inception of the Robust track was motivated by the failures of retrieval systems on difficult topics. In order to encourage researchers to improve the retrieval technology without severely degrading performance, especially on these difficult topics, the robust track (Robust) was started in 2004. This task emphasized improving system effectiveness consistently, across a mixed set of topics that were used in this task, consisting of a subset of difficult topics, that is, low scoring but with some systems performing well. The underlying document collections remain unchanged, which is similar to the ones used in the TREC-8 Ad Hoc track.

With the development of web search techniques and the requirement of designing new retrieval methods for specific information needs, more tracks have been added in later TREC conferences. One of the most widely used tracks for testing both effectiveness and scalability is the web track, starting from TREC-9. The first web track focused on examining the impact of link information on effectiveness. Later web tracks included the Ad Hoc track as well, but with larger test collection (web pages) than earlier Ad Hoc track. Subsequent web tracks also included tasks such as diversification, extending beyond ad hoc retrieval.

2.5.2 Topics and Judgments

The previous section mainly introduced the document organization in the context of Ad Hoc tracks, and for this section we detail both the TREC topics and judgment process.

The TREC Topics. Each task typically contains 50 topics, and each topic represents a user's information need, including different topic aspect, as shown in Figure 2.8(b). Unlike a user query, the TREC topic reflects a user's information need, which is described using all the fields listed. The provided topic title is often referred to as the "title query" in a retrieval task; other fields have often been used to provide alternative inputs to a retrieval system. In early TREC Ad Hoc tasks, a

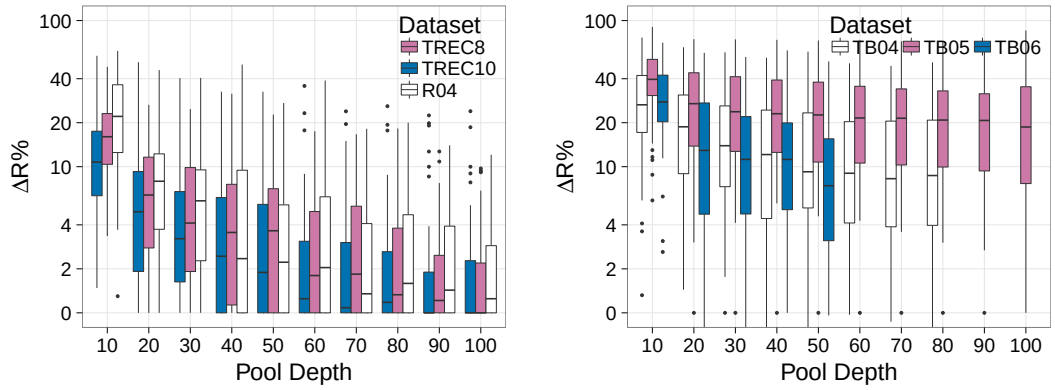


Figure 2.9: Observations for different pooling statistics on two types of TREC collections. The $\Delta R\%$ is the percentage of relevant documents found when an additional set of documents are judged. Both plots show $\Delta R\%$ as a function of the pooling depth, using $\log$ scale. The panel on the left shows results on earlier TREC tasks and the panel on the right shows the observation on three GOV2 test collections.

topic is created by a system user who is also the assessor responsible for judging relevance of retrieved documents. Recent tracks, such as web tracks, often make use of samples from web search logs to create topics [26]. The TREC topics also have different definitions based on specific tasks, for example, in a diversification task, a main topic is associated with several sub-topics derived from an interpretation of the main topic. Each sub-topic may be associated with a predefined category, and this has become more widely adopted in recent test collections, for example, in the TREC 2009 web track, a sub-topic could be either navigational or informational.

Judgments. For each topic, a set of documents are selected to be judged. Typically, a top-d pooling process is often used to form the judgment pool, as described in Section 2.4.1. Two exceptions were the TB06 and CW09A-2009 test collections, where a sampling procedure operated in addition to the top-d judgment pool. The deep judgment pools on small test collections were able to identify the majority of relevant documents so that systems could be fairly evaluated using these test collections. Therefore, there is less bias in the evaluation process, when assuming unjudged documents are not relevant [134]. Nevertheless, although a deep pool has been formed for TeraByte tasks, due to the large size of the document collection, the judgment pool still does not achieve a high coverage of the relevant documents for each topic. The different coverage of relevant documents on the two types of test collections are plotted in Figure 2.9. Both panels in Figure 2.9 show the incremental percentage of relevant documents found as a function of pooling depth. When considering existing TREC judgments, it is obvious that only a small percentage of relevant documnets can be further found when increasing the pooling depth, on TREC-8, TREC-9 and the Robust04 test collection. Of the three TeraByte tasks, a roughly similar percentage of relevant documents can be found during each incremental step of pooling depth. If the same extrapolation approach used by Zobel [134] is applied to the three TeraByte tasks, a pooling depth much deeper than 100 would be required in order to obtain a judgment pool with high cover-

Tasks	Corpus	Size (cmp.)	#Docs	d	n
Robust04	Disc 4 &5‡	1904 MB	0.5 Million	100	56
TB04	GOV2	426 GB	25 Million	85	34
TB05				100	36†
TB06				50*	40†
CW09-2009	ClueWeb09A (B)	5 TB (A) 500 GB (B)	1 Billion (A) 50 Million (B)	12*	32†
CW09-2010				20	21†
CW09-2011				25	23†
CW09-2012				20*	13†

Table 2.6: An overview of the test collections according to the TREC reports. The column "Tasks" lists each test collection in "track-year" form. The "Corpus" column lists the document collections used for tasks. The last two columns "d" and "n" are the TREC pooling depth and contributing systems, respectively.

age of relevant documents. On the ClueWeb document collection, a shallower pooling depth was employed compared to the three TeraByte test collections. As can be inferred from Figure 2.9, the judgment set is even more incomplete when compared to the TeraByte test collections, and raises important questions in connection with default evaluation settings that assume unjudged documents are not relevant. Further, as the evaluation process may be conducted to any arbitrary rank position instead of over the entire ranked list, understanding the impact of different "evaluation depth (k) and pooling depth (d) and evaluation metric (M@k)" configurations is a critical component when interpreting system effectiveness scores.

2.5.3 Summary of Test Collections

A summary of test collections used in the experiments in the remainder of the thesis are listed in Table 2.6. The column "Corpus" lists the document collection used for the corresponding tasks; d is the pooling depth and n is the number of systems that contributed to the judgment pool. For CW09-based tasks from 2009 to 2012, all submitted systems are used in the pooling process. A "†" indicates that for the number of contributing systems, we listed only the number of Ad Hoc runs only, regardless of runs from other tasks. A "$*$" indicates that the listed pooling depth is the minimum depth used, while there may be more judged documents pooled from deeper ranks for some runs. The "‡" indicates that the Robust04 test collection makes use of part of the documents listed in the table. Note that for CW09-2009 to CW09-2012 tasks, there are two document collections available. They are differentiated using a suffix "A" or "B" in experiments.

2.6 Summary

Chapter 2 has provided an overview of the field of information retrieval, with emphasis on two parts, the proximity-based retrieval and collection-based system evaluation. In Section 2.2, we introduced heuristics for building information retrieval models and the relationships between a user query and documents was established based on those heuristics. Starting with various features

derived from retrieval heuristics, which are widely applied in more advanced retrieval models, we then proceeded to describe proximity features from a categorical perspective. Based on different types of statistics used, we introduced several proximity-based retrieval models and summarized the findings of effectiveness gains when using these proximity-based retrieval models. Nevertheless, the computation cost of proximity features was found to be high and thus we proceeded by covering the representations of documents adopted in existing systems (Section 2.3). We found that, while various representations were proposed in the literature, they are inadequate at improving the efficiency of proximity-based retrieving models, especially when higher-order proximity features are to be considered. For example, the proposed representations may consider phrase-based proximity or only term-pair proximities, but they were not good for generalization. Therefore, we consider a generalized version of improving proximity feature computation in Chapter 3 and show that we can make good use of proximity statistics in Chapter 4.

In this chapter, we also introduced the widely applied batch evaluation methodology, including the building of test collections and classic evaluation metrics. However, when a large-scale experimental setting is required and web-scale test collections are used, the offline evaluation suffered from different types of bias since only a limited judgment budget is available. Section 2.4.4 introduced several methods that aimed at providing reliable system comparison results. Whilst the proposed methods could mitigate the problem to some extent, most of them were designed for one type of bias – system bias. A few methods can reduce some pooling depth bias, but none fully resolve it. One of the concerns of researchers related to the pooling depth bias was in the extended evaluation process, since all classic evaluation methods were required to be used in a truncated computational format. Chapter 5 provides a detailed experimental study on the impact of choosing the evaluation and pooling depth. Finally, in order to reliably compare systems, score adjustment methods that focused on estimating true performance of systems have been studied (Section 2.4.4). We consider this problem in Chapter 6, aiming to reduce pooling depth bias.

CHAPTER 3

Computing Term Proximity Features

As already stated in Chapter 2, there is an extensive body of literature on proximity-based retrieval models [17, 24, 42, 43, 73, 82, 119]. Empirical evidence suggests that retrieval models based on proximity features are often more effective than their bag-of-words counterparts [73]. Proximity-based retrieval methods go beyond unigram models by considering the relationships between terms in a document. One of the most widely used retrieval models which captures term-dependencies using Markov Random Fields (MRF) was proposed by Metzler and Croft [73]. There are two variants of the MRF-based proximity ranking model – the Sequential Dependency Model (SDM) and the Full Dependency Model (FDM). Both consider two different types of proximity features, ordered and unordered windows. SDM considers proximity between two query terms while FDM considers all possible relationships among query terms. A recent empirical study by Huston and Croft [45] showed that on some test collections, both SDM and FDM are significantly more effective than bag-of-words retrieval models. However, it is unclear as to which proximity features offer the most gain in terms of effectiveness and to what extent such features impact computational cost.

Section 2.2.2 described several possible categories for proximity retrieval methods. Term-dependency retrieval models can be categorized in a number of ways, either by the proximity features they use or by the underlying model itself. For instance, consider the family of MRF retrieval models for which the property of ordering, or lack thereof among the terms, is one form of categorization. When considering MRF-based proximity models, there are two ways of categorizing features used in the models. When classifying based on the number of query terms considered in the term-dependencies, Huston and Croft [45] compare FDM and SDM, and conclude that using proximity features calculated between two terms is often sufficient for improving retrieval effectiveness. The alternative categorization is based on whether the term-dependencies require terms to appear consecutively or not. This can be explored by considering each component leveraged within one retrieval model, as shown in Equation 2.6. An *ordered window* requires the n query terms to appear consecutively within a specified distance, and an *unordered window* only has the distance restriction, regardless of which order the n terms appear. Table 2.2 suggests researchers have concentrated on the cost of computing term-pair proximity features or ordered

window features, and that little attention has been paid to the efficient computation of proximity features consisting of unordered windows that capture more than two terms. This may be due to the exponential number of possible term dependencies that makes offline computation expensive. The first online solution of computing higher-order term-dependencies was proposed by Sadakane and Imai [92], which was based on the PLANESWEEP algorithm.

The PLANESWEEP method [92] computes the unordered window proximity features of a single query efficiently. However, as is indicated by the name, the FDM model considers all possible relations among query terms, which requires proximity features to be extracted for multiple subqueries, resulting in a significant computational overhead. Therefore, it is important to explore the cost of computing proximity features, and more specifically, unordered proximity features.

This chapter is laid out as follows. We begin in Section 3.1 by introducing the background of higher-order proximity features, their definitions and the PLANESWEEP extraction method proposed by Sadakane and Imai [92]. Section 3.2 examines the effectiveness of different types of proximity features in the MRF family of term-dependency models. This prior work provides motivation for efficiently computing higher-order proximity features. In Section 3.3, we consider the cost of computing higher-order proximity features on a fixed set of terms, and two different input representations. Section 3.4 extends the examination of proximity features pertaining to FDM, where multiple sets of terms are considered for extracting proximity features. The chapter concludes with Section 3.5, which explores the effectiveness of different proximity features used in MRF-based proximity models, and the different costs of computing the higher-order proximity features using proposed methods.

3.1 Background

3.1.1 Subqueries, Optimal Intervals and Fixed Size Windows

Optimal Intervals. Before making any further distinctions between proximity features, we consider the definition of the *optimal* $|Q|$-interval that captures all query terms in a query Q. Let D be a document, and $D\,[p_\ell \ldots p_r]$ a sequence of terms bounded with left position p_ℓ and right position p_r. Then an optimal interval is:

Definition 1. (Optimal Interval). An interval $[p_\ell \ldots p_r]$ bounded by left position p_ℓ and right position p_r is optimal to a query Q if all terms in Q appear in $D\,[p_\ell \ldots p_r]$, and there is no proper subset of $D\,[p_\ell \ldots p_r]$ with the same property.

Consider the document shown in Figure 3.1, and an example subquery Q'_3: "`scalable vector graphics`". There are two intervals that contain all three query terms. A similar idea was proposed and adopted by Clarke et al. [24]. According to Definition 1, only $D\,[0 \ldots 2]$ is optimal, since it is a proper subset of $D\,[0 \ldots 11]$. In some retrieval models such as FDM, intervals are overlapped and are often referred to as a "window". We will use "window" when describing the intervals used in such models. Otherwise, we will use "interval" to refer to proximity features in general.

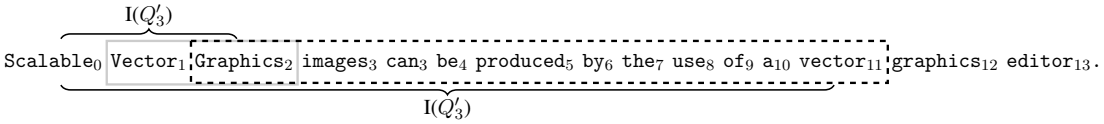


Figure 3.1: Interval examples, the two curly brackets are intervals for the subquery "`scalable vector graphics`". The two rectangles are intervals for the subquery "`vector graphics`".

Subqueries. Proximity features are query dependent, and often represented as frequencies of intervals capturing query terms. These intervals can be overlapping or non-overlapping, depending on how the retrieval model defines the dependencies among query terms. Usually, aside from the original terms issued by users, a set of *subqueries* are considered by the retrieval model representing relations beyond a bag-of-words representation. Take for example the query "`scalable vector graphics`", and let Q'_l be a subquery containing l query terms from the original query. For example, one of the two-term subqueries of the example query is $Q'_2 = \{$"`scalable graphics`"$\}$. Let $\mathcal{Q}'_l$ be the set of l-term subqueries, and for any original query Q, $|\mathcal{Q}'_l| = \binom{|Q|}{|l|}$. Recall the MRF-based term-dependency model and consider the full dependency model, which scores a document given a query Q as:

$$\begin{aligned} \mathit{Score}(Q, D) = \lambda_t \cdot \sum_{q \in Q} \mathit{Score}(q, D) \\ + \lambda_{ow} \cdot \sum_{l=2}^{|Q|} \sum_{Q'_l \in \mathcal{Q}'_l} \mathit{Score}(I_{ox}(Q'_l), D) \\ + \lambda_{uw} \cdot \sum_{l=2}^{|Q|} \sum_{Q'_l \in \mathcal{Q}'_l} \mathit{Score}(I_{ux}(Q'_l), D), \end{aligned} \tag{3.1}$$

where $I_{ox}(I)$ and $I_{ux}(I)$ are ordered and unordered windows with a user-defined fixed length denoted by x. SDM differs from the FDM model by the set of subqueries considered; SDM considers only $Q'_l \in \mathcal{Q}'_2$, ignoring all the other higher-order subqueries, $Q'_l(l > 2)$. Both ordered and unordered windows are considered over the set of subqueries, with the only difference being whether the terms within a considered subquery are consecutive or not. Ordered windows are computed over the set of subqueries whose query terms are consecutive, in other words, phrases. Thus the set of subqueries considered as the basis of computing ordered windows are $\mathcal{Q}'_2 = \{$"`scalable vector`", "`vector graphics`"$\}$ and $\mathcal{Q}'_3 = \{$"`scalable vector graphics`"$\}$. Unordered windows relax both order and consecutive requirements when considering proximity features and are computed over all possible combinations of query terms. For example, in order to score the example query, two sets of subqueries $\mathcal{Q}'_2 = \{$ "`scalable vector`", "`scalable graphics`", "`vector graphics`"$\}$ and $\mathcal{Q}'_3 = \{$"`scalable vector graphics`"$\}$ are considered. For a query of length l, FDM considers $(l+2)(l-1)/2$ subqueries in order to compute the ordered window features and $2^l - l - 1$ subqueries for calculating unordered window features.

Fixed Size Windows. As aforementioned, MRF-based term-dependency models consider two types of proximity features – ordered and unordered windows. An ordered window is created when

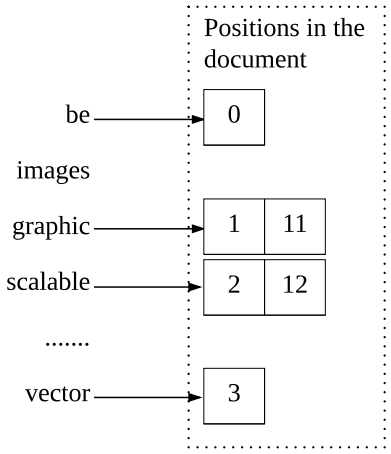


Figure 3.2: An example of position lists of terms in the document shown in Figure 3.1.

the query terms appear in the same order as they occur in the considered subquery. Unordered windows are created without this restriction. In addition to the windows in the document, the set of subqueries considered by the two features are also different. Ordered windows are extracted in accordance with subqueries that are n-grams of the original query, while unordered windows are not. Take for example the two optimal intervals as shown by the two rectangles in Figure 3.1. When the subquery is "`vector graphics`", and the ordering constraint is imposed, then the only candidate for consideration is $D\,[1\ldots2]$. This is because the query terms appearing in the interval $D\,[2\ldots11]$ are not in the same order as the query. Only when an unordered window proximity component is examined, do both intervals become optimal for "`vector graphics`". FDM constructs a term dependence model using the proximity features discussed within a set of subqueries, which are then derived from the original query. For FDM, this is the proximity features of all $2^{|Q|} - |Q| - 1$ subqueries. It is these properties that may help models such as FDM to exhibit greater effectiveness than bag-of-words models. The rapid growth as a function of $|Q|$ implies that methods for computing intervals over all proximity statistics in a single-pass are of particular interest. This produces the following problem that we now consider: *for document* $D[1\ldots N]$ *and query* $Q = \{q_1, q_2 \ldots, q_k\}$, *find all optimal* $|Q'|$*-intervals in* D *for all non-empty subqueries* $Q' \subseteq Q$ *in which* $|Q'| \geq 2$. Consider again the example query "`scalable vector graphics`". The problem is to identify all optimal intervals for all subqueries in the example document shown in Figure 3.1. These are: $D\,[0\ldots1]$ for the subquery "`scalable vector`", $\{D\,[1\ldots2]\,,\ D\,[2\ldots11]\,,\ D\,[11\ldots12]\}$ for the subquery "`vector graphics`"; and $D\,[0\ldots2]$ of subquery "`scalable vector graphics`".

It is worth noting that when integrating the optimal intervals into the FDM model, a predefined distance is required. The intervals satisfying the distance requirement are often referred to as *fixed size* windows. In this thesis, we use $I_{ux}(Q')$ and $I_{ox}(Q')$ to represent ordered and unordered windows with size x, respectively.

3.1.2 Representations for Finding Optimal Intervals

In Chapter 2, we have seen two main representations of positional information exploited in IR systems: *inverted files* and *direct files*. Both the inverted files and direct files are considered in offline preprocessing steps, independent of user queries. When user queries are taken into consideration, there is a third option – a Condensed Direct File (CDF). It is defined as:

Definition 2. (Condensed Direct File (CDF)). A condensed direct file is the direct file containing only the query terms for a particular given query Q, stored in the order they appear in the document.

The condensed direct file of the document shown Figure 3.1, and query "`scalable vector graphics`" is "$\texttt{scalable}_0$ $\texttt{vector}_1$ $\texttt{graphics}_2$ $\texttt{vector}_{11}$ $\texttt{graphics}_{12}$". A CDF can be generated from either inverted files or an existing direct file, by taking only query terms into consideration. Note that since a CDF is query dependent, it cannot be obtained during a pre-computation step and it is an intermediate representation produced at query evaluation time.

The best choice of input representations for finding optimal intervals is not clear. Both inverted files and direct files can be applied in the computational process. For example, the Terrier search engine uses direct files when computing similarity scores using proximity-based retrieval models, while the Indri tool only evaluates queries using inverted files, regardless of the type of retrieval model used. Although Asadi and Lin [3] concluded that it is more efficient to perform feature extraction using a direct file representation, there is no evidence this is the best representation during query evaluation, or how different representations might affect the proximity calculation process. We explore answers to these questions in this chapter.

3.2 Higher-Order Proximity Features in Term-Dependency Models

Section 3.1 introduced the proximity features used in term-dependency models, and discussed the cost of computing proximity feature statistics for all subqueries. Before considering solutions to extracting proximity features, it is necessary to understand the role of different proximity features in term-dependency models. There have been previous studies on the difference between term-proximity features for subqueries Q'_2 and $Q'_{\geq 2}$ [45, 111], but few studies have compared the impact of using ordered and unordered windows independently. This section focuses on exploring the impact of ordered and unordered features in MRF-based term-dependency models.

3.2.1 Types of Proximity Features in Term-Dependency Models

As shown in Equation 3.1 and based on whether order is required, proximity features in the model fall into two types. In order to explore the role of each type of feature in improving model effectiveness, the configurations listed in Table 3.1 were explored. The set of configurations are derived based on three components in the term-dependency models – a bag-of-words (BOW) component, an ordered window (OW) and an unordered window (UW). As mentioned before, ordered windows require the subquery to be an n-gram of the original adjacent query terms. Note that in

Model	Features	Subqueries	Configurations
BOW	unigrams	individual $q \in Q$.	$\lambda_t = 1$.
BOW + OW	unigrams $I_{ox}(Q')$ with $x = 1$.	adjacent, $\|Q'\| \geq 2$.	$\lambda_t = 0.85$, $\lambda_{ow} = 0.15$.
BOW + UW	unigrams; $I_{uy}(Q')$ with $y = 4 \cdot \|Q'\|$.	all $Q' \in Q'_i\,(i \geq 2)$.	$\lambda_t = 0.85$, $\lambda_{uw} = 0.15$.
SDM	unigrams; $I_{ox}(Q')$ with $x = 1$; $I_{uy}(Q')$ with $y = 8$.	adjacent for all $Q' \in \mathcal{Q}'_2$.	$\lambda_t = 0.8$, $\lambda_{ow} = 0.1$, $\lambda_{uw} = 0.1$.
FDM	unigrams; $I_{ox}(Q')$ with $x = 1$; $I_{uy}(Q')$ with $y = 4 \cdot \|Q'\|$.	all subqueries $Q' \in \mathcal{Q}'_i$, $(i \geq 2)$.	$\lambda_t = 0.8$, $\lambda_{ow} = 0.15$, $\lambda_{uw} = 0.05$.

Table 3.1: Configurations of term-dependency models explored. The "Model" column lists different types of models considered in the comparison study; the "Features" column lists the types of intervals considered in the model; the "Subqueries" column indicates the set of subqueries, based on which the set of proximity features are extracted; the "Configurations" column lists default parameter values of considered features.

SDM, the set of subqueries leveraged by the model is restricted to $\mathcal{Q}'_2$, instead of all $2^{|Q|} - |Q| - 1$ combinations.

The question "*how effective are proximity features in term-dependency models*?" is considered by exploring each type of proximity feature individually. A bag-of-words model that considers only unigram features is used as the reference system. Our hypothesis is that, if a type of proximity features is important, then it should improve the effectiveness compared to a bag-of-words model, or at least, this type of feature should not degrade the performance.

3.2.2 The Impact of Proximity Features in Term-Dependency Models

Datasets, Queries and Evaluation Metrics. All experiments were performed using Indri, Krovetz stemming, and dependency models generated using Metzler's Indri configuration[1]. We keep stopwords in our index since it may have an impact when considering phrase features. Three different TREC test collections and topic sets were used: Robust04, TB04–06 and CW09A-2009/2011/2012. Details of these test collection were listed in Table 2.6. Note that we use a pruned ClueWeb collection in which only documents with a spam score greater than 70 are included. This greatly improves the quality of results for all models tested. See Cormack et al. [30] for further information. For each query, the experiments return the top 1,000 documents. Results are evaluated using $\text{NDCG}_b@10$, $\text{RBP}(0.95)$; and $\text{AP}_a@1{,}000$, defined in Chapter 2. Results are then averaged over query sets, and query-by-query scores used for statistical testing.

[1] http://ciir.cs.umass.edu/~metzler/dm.pl

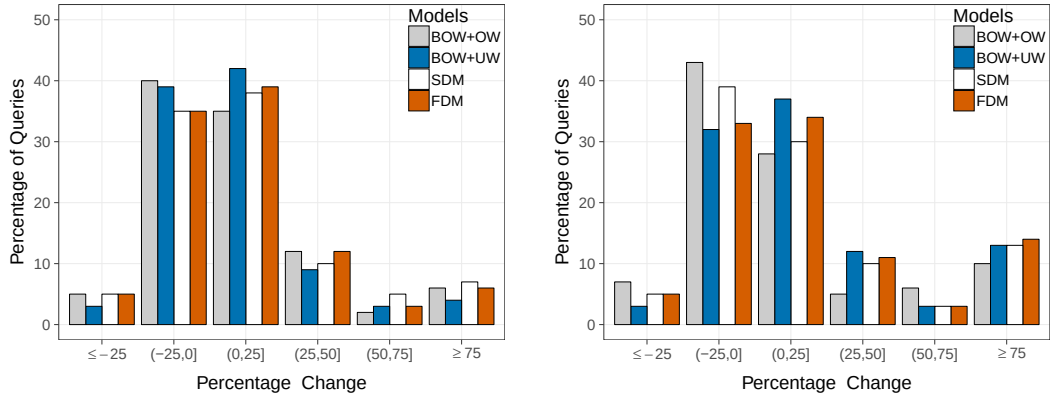


Figure 3.3: Distribution of relative query performance deltas of models listed in Table 3.1, categorized by delta size. The evaluation metric used is AP. The first figure shows results for the Robust04 test collection, with 238 queries considered, and the second one shows results for the TB04–06 test collection using 147 queries. Single-term queries are not considered in this study.

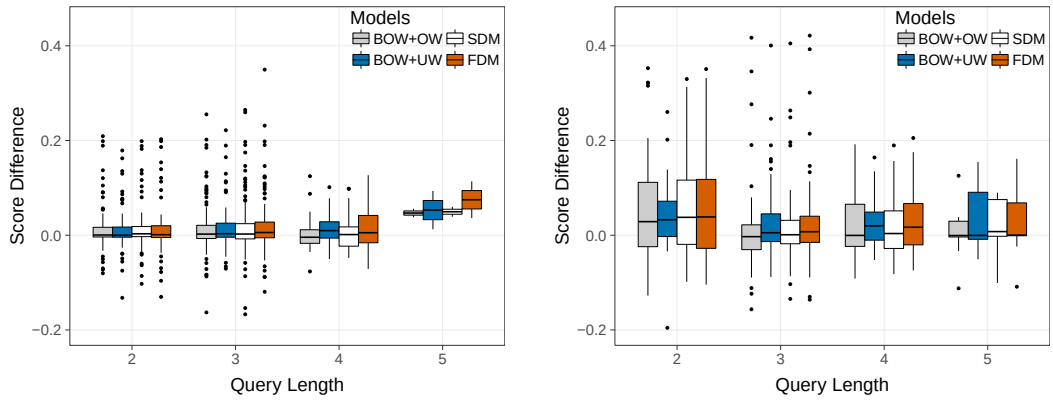


Figure 3.4: Score differentials relative to the bag-of-words model, broken down by query length. All the experimental configurations are the same as Figure 3.3. The first figure shows results on the Robust04 test collection and the second figure shows results on the TB04–06 test collection.

Observations on the Effectiveness Changes. Table 3.2 lists measured effectiveness scores for all five retrieval models and three test collections. All of the dependency models increase overall effectiveness in general. The interesting trend is that BOW+UW is always better than BOW+OW, and has very similar performance to SDM and FDM. In fact, BOW+UW is marginally better than SDM for all metrics on the TB04–06 test collections. The AP and RBP scores show that BOW+UW can achieve better effectiveness than BOW+OW on the CW09A tasks. However, we also note the abnormal results on the CW09A collection compared to the other datasets. This may be caused by using the default configuration of weighting parameters which were tuned by Metzler on much smaller datasets. A similar observation was also made by Vuurens and de Vries [119]. As suggested by Metzler and Croft [73], using an unlimited window size with SDM or FDM performs better on large collections, which could also be the reason. Despite using an out-of-the-box configuration, most of the results are statistically significant with the exception of BOW+OW

Model	Robust04			TB04–06			CW09A-09 & 11–12		
	NDCG	RBP	AP	NDCG	RBP	AP	NDCG	RBP	AP
BOW	0.4366	0.3080	0.2477	0.4396	0.4775	0.2920	0.2110	0.2553	0.1168
BOW+OW	0.4512†	0.3192	0.2599	0.4603	0.4940	0.3121	0.2266	0.2708	0.1248
BOW+UW	0.4522†	0.3201†	0.2606†	0.4892†	0.5126†	0.3221†	0.2224	0.2729†	0.1278†
SDM	0.4557†	0.3218†	0.2618†	0.4781†	0.5062†	0.3193†	0.2284	0.2777†	0.1299†
FDM	0.4512†	0.3223†	0.2635†	0.4870†	0.5119†	0.3265†	0.2308	0.2778†	0.1292

Table 3.2: Effectiveness of ranking models for Robust04, TB04–06 and CW09A-09&11-12. Values of the model parameters are listed in Table 3.1. When evaluated using NDCG, the evaluation depth is 10 and all the other metrics are evaluated to the full depth (1,000). The persistence value $p = 0.95$ is used for RBP. For RBP the residual (corresponding to unjudged documents appearing in the rankings) were all less than 0.04 for Robust04 and TB04–06; but between 0.34 and 0.35 for CW09A tasks. A superscript † suggests a performance difference with significance $p \leq 0.05$ when compared to the baseline the bag-of-words runs, calculated using a paired t-test.

which failed to show a significant improvement over bag-of-words runs for larger collections.

Also worth noting in connection with Table 3.2 are the high RBP residuals for the CW09A experiments. The high residual is evidence that large numbers of unjudged documents are being retrieved in the runs, suggesting the measured scores are not precise. With a residual that is of similar magnitude to the actual scores, indicating the CW09A results need to be treated with caution. The AP scores are also likely to be affected by this issue, but the extent of this effect is not easily quantified. It is also important to know that, although NDCG$_b$@10 is used in this experiment, the shallow judgment pool may also affect the reliability of evaluation results. The problem of reliable evaluation on large collections with shallow judgment pools will be further discussed in Chapters 5 and 6. As the results on CW09A test collections may be biased, we then focus on the performance comparison of the five models on Robust04 and TB04–06.

Figure 3.3 shows the performance measured using AP, relative to the bag-of-words baseline, the models compared are: BOW+OW, BOW+UW, SDM, and FDM. The first pane shows the comparison results on Robust04 and the second shows results on TB04–06. The x-axis group relative performance differences, and the y-axis shows the percentages of queries in the corresponding group. Note that proximity features are the focus, so single-term queries are not considered. As expected, all of the enhanced bag-of-words models improve the results for many queries, although they do reduce the effectiveness of some queries. Perhaps the most noticeable outlier is the ordered window component. In both collections it causes a $25{+}\%$ degradation for several queries. Since the SDM and FDM methods also incorporate an OW component, albeit with a reduced weighting, they also experience a degradation in performance. In contrast, the unordered window feature alone causes less queries to suffer from more than 25% performance degradations compared to all the other three models. This suggests that using proximity features without an order constraint may be more robust.

The breakdown by query length in Figure 3.4 shows a similar trend. The BOW+UW variant has a few negative outliers but in general, dependence models using OW components are more

Query	Dataset	Model	ΔAP
"literacy rates Africa"	Robust04	BOW+OW	–0.0501
		SDM	–0.0478
		FDM	–0.0076
"legal pan am 103 "	Robust04	BOW+OW	–0.0341
		SDM	–0.0167
		FDM	–0.0170
"southeast Asia tin mining"	Robust04	BOW+OW	–0.0227
		SDM	–0.0332
		FDM	–0.0127
"1890 census"	TB06	BOW+OW	–0.0807
		SDM	–0.0472
		FDM	–0.0528
"kyrgyzstan-united states relations"	TB04	BOW+OW	–0.0552
		SDM	–0.0548
		FDM	–0.0260
"schizophrenia drugs"	TB05	BOW+OW	–0.0425
		SDM	–0.0381
		FDM	–0.0412

Table 3.3: Top three outlier queries identified for the Robust04 and TB04–06 collections, sorted by $|\Delta AP|$ of OW. These queries suffer from degraded effectiveness in models containing a OW component relative to the BOW, but not on the BOW+UW model.

likely to incur a significant degradation in effectiveness for certain queries.

Table 3.3 shows the three queries which are hurt most by the OW component, but benefit from using a UW component alone. These queries exemplify the pitfalls of presuming that all contiguous query subsequences are "good" phrases. Brute force partitioning of "legal pan am 103" results in fragments such as "legal pan" and "am 103". These fragments are likely to have an unexpectedly high impact, whilst not being relevant to the query. On the other hand, it is a little surprising that treating "schizophrenia drugs" as a phrase component hurts retrieval effectiveness. Query partitioning is a difficult problem even when done manually, and these results show that there remains a lot to learn about automatic query parsing. Table 3.4 shows the top three outliers for each test collections identified using the effectiveness loss of BOW+OW model. It shows that although all models suffer from degraded effectiveness to some extent, the model BOW+UW shows a more robust behavior, except for the query "toxic chemical weapon". Even by considering a combination of both OW and UW, the FDM model also suffers from a similar effectiveness loss. Also, it is interesting to note that applying the proximity heuristic to query "iceland government" hurts retrieval effectiveness.

Conclusions in Regard to the Effectiveness Impact of Using Proximity Features. This section investigated the relative effectiveness of proximity in dependency-based ranking models. The experiment results suggest that considering proximity features without order restrictions may be

Query	Dataset	Model	ΔAP
"timber exports Asia"	Robust04	BOW+OW	–0.1629
		BOW+UW	–0.0706
		SDM	–0.1670
		FDM	–0.1194
"falkland petroleum exploration"	Robust04	BOW+OW	–0.0839
		BOW+UW	–0.0659
		SDM	–0.0719
		FDM	–0.0883
"toxic chemical weapon"	Robust04	BOW+OW	–0.0227
		BOW+UW	–0.0332
		SDM	–0.0332
		FDM	–0.0127
"iceland government"	TB06	BOW+OW	–0.2402
		BOW+UW	–0.1955
		SDM	–0.2173
		FDM	–0.2539
"scottish highland games"	TB05	BOW+OW	–0.1564
		BOW+UW	–0.0703
		SDM	–0.1344
		FDM	–0.1300
"iran terrorism"	TB04	BOW+OW	–0.1273
		BOW+UW	–0.0037
		SDM	–0.0984
		FDM	–0.1042

Table 3.4: Top three outlier queries identified for the Robust04 and TB04–06 collections, sorted by $|\Delta AP|$ of BOW+OW.

more robust than their ordered counterparts; and that a combination of both ordered and unordered proximity features may be better and more robust than considering individual features. However, care must be taken when incorporating ordered phrase components, as the results suggest that the effectiveness of some queries can be significantly degraded for reasons that are difficult to anticipate. It remains unclear how to automatically identify the set of subqueries on which to apply proximity features to reduce the likelihood of degraded performance in outlier queries. Recent work on dependency models has focused on selective weighting of concepts in queries [10]. This has not been investigated in this work, but it is clear that selective weighting based on the query terms has the potential to improve effectiveness, across all aspects of dependency model processing. How the different choices of local, global and approximate proximity statistics will change in terms of retrieval effectiveness has also been explored by Macdonald and Ounis [66], and this is explored in Chapter 4.

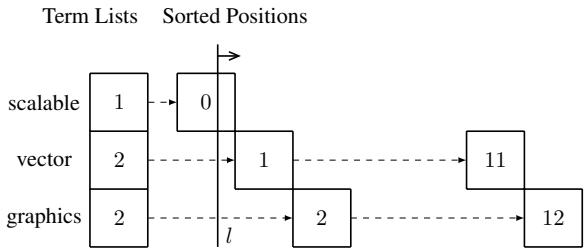


Figure 3.5: The PLANESWEEP algorithm checks whether the interval formed by the current leftmost representative term and its next appearance encloses all other query term endpoints. A virtual line sweeps from left to right, with term positions determined via an inverted file.

3.3 Extracting Single Intervals

As discussed in the previous section, unordered proximity features may be more robust than ordered ones. Therefore, it is necessary and important for us to consider the problem of computing unordered windows. There is a rich literature on computing optimal intervals for subqueries consisting of two query terms in both an offline and online manner [13, 35], and were discussed in Chapter 2. However, when a subquery contains more than two query terms, it is difficult to give an exact offline solution, especially when there is no order-preserving requirement on the set of candidate intervals. Asadi and Lin [3] suggest using an adaptive set intersection method for computing optimal intervals. However, their work focused on two-term subqueries. How to extract optimal intervals with regard to a given query has seen less attention. This section is mainly connected with the problem of finding optimal $|Q'|$-intervals, for a given subquery Q'. Throughout, we consider processing a query Q relative to a document D.

3.3.1 The Eager PLANESWEEP (EPS) Approach

An approach that considers optimal interval extraction for a given subquery Q' beyond term-pairs was presented by Sadakane and Imai [92]. The set of optimal intervals of a given query Q' is extracted from using a PLANESWEEP method. The PLANESWEEP approach processes terms in the order they appear in the document, checking whether the interval formed by the current leftmost term encloses all of the other endpoints. For example, consider the scenario shown in Figure 3.5. When the query is "`vector graphics`", the algorithm checks the next position of the current leftmost query term "`vector`" at position 11. Since the occurrence of "`vector`" at position 11 is greater than the position of the current rightmost term "`graphics`" occurring at position 2, the algorithm then determines that $[1\ldots2]$ is an optimal interval. The process continues by considering "`vector`" again as the leftmost term since $11 < 12$, and because both positions are the last ones of the two considered query terms. The algorithm then terminates returning the two intervals identified via the sweeping process. Extracting optimal intervals for three-term queries works similarly as the process described for the query "`vector graphics`". Using the query "`scalable vector graphics`" as an example, the current leftmost term "`scalable`" occurs only once and the current interval $[0\ldots2]$ is the only 3-optimal interval for the current query.

The PLANESWEEP approach can be visualized as sweeping a virtual line from left to right across term positions within a document, hence the name. If the positions are represented using positional postings lists, then the next leftmost representatives are easily selected as are their successors, and all of the required information can be kept up to date with relatively little effort. For reasons that will become apparent shortly, we refer to this *eager* version of PLANESWEEP. Algorithm 1 presents it in detail.

Algorithm 1 Eager PLANESWEEP, Single Interval (EPS-IF-S)

Input: A query Q of k terms, and sorted position lists of all terms $\{P_1, P_2, \ldots, P_k\}$ in document D

Output: A set of intervals $I(Q)$

```
    I(Q) ← {}
    for i ← 1 to k do
        curr_occ[i] ← FIRST(P_i)
    end for
5:  while (max_{1≤i≤k} curr_occ[i]) < ∞ do
        lsym ← argmin_{1≤i≤k}{curr_occ[i]}
        rsym ← argmax_{1≤i≤k}{curr_occ[i]}
        lpos ← NEXT(P_lsym)                    // returns ∞ if P_lsym is exhausted
        if lpos > curr_occ[rsym] then
10:         I(Q) ← I(Q) ∪⟨curr_occ[lsym]...curr_occ[rsym]⟩
        end if
        curr_occ[lsym] ← lpos
    end while
    return I(Q)
```

The inverted index is presumed to support the two usual access operations: FIRST and NEXT. The operation FIRST returns the first position of a term in the document being examined and the operation NEXT returns the next position of a term, both of which are operated on the current document being processed. The $|Q|$-element array *curr_occ* is used to track the current positions of all terms in the query. In addition, *lpos* maintains the next position of *lsym*. If *lpos* is greater than *curr_occ*[*rsym*], the current interval is emitted. Finally in each iteration, *curr_occ*[*lsym*] is updated to the new most recent position, *lpos*. When any of the position lists is exhausted, the algorithm terminates. Figure 3.6 gives a concrete example of this process, matching the schematic shown in Figure 3.5.

Since each of the postings lists in the inverted index is already sorted by term positions, Algorithm 1 requires $O(\log |Q|)$ time per iteration, using two heaps of $|Q|$ items to identify maximal and minimal representatives at steps 6 and 7 respectively. All other operations require $O(1)$ time each. Therefore, over the $|Q|$ terms, there is a total of $T_{Q,D} = \sum_{q \in Q} f_{q,D} \leq |D|$ postings to be processed, and the total execution time is $O(T_{Q,D} \log |Q|) = O(|D| \log |D|)$ [92].

3.3.2 The Lazy PLANESWEEP (LPS) Approach

In the EPS approach discussed in Section 3.3.1, the leftmost position is used as a pivot, and the next occurrence of that term is used to determine if the pivot is the leftmost end of an interval.

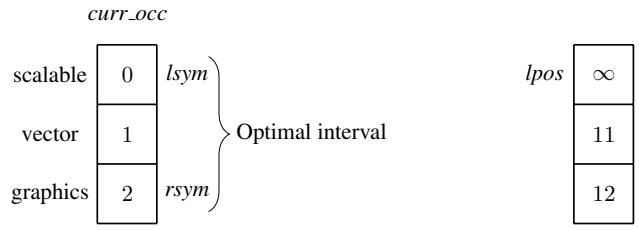


Figure 3.6: A running example of finding optimal intervals for the query "`scalable vector graphics`" using the EPS approach. The current *lsym* is "`scalable`", since its position is the smallest among all in *curr_occ*. The current *rsym* is "`graphics`" because its position is the maximal value among all three terms. The *lpos* is ∞ because positions of *lsym* have been exhausted, and thus $[0\ldots2]$ is an optimal interval for the current query.

However if the input format is a direct file, the NEXT operation cannot be carried out in $O(1)$ time, and eager updating of *curr_occ* has the potential to induce a much higher computational cost. When the input representation is in the form of a direct file, instead of checking the next position of the leftmost term, it is possible to check whether an optimal interval is found when the right-hand extrema is processed. As with the EPS implementation, the most recent occurrence of each term is tracked in *curr_occ*. A sentinel value of $-\infty$ is used to indicate the absence of a term, and an interval can be formed only when a usable occurrence of each term is defined within the scope. That is, intervals are identified and emitted when the right-hand end of the interval is reached. If an interval is identified, the smallest of the values in *curr_occ* is then voided, providing a signal that the next interval must contain a new instance of that term.

Algorithm 2 Lazy PLANESWEEP, Single Interval (LPS-DF-S)

Input: A query Q, and a direct file $D[1\ldots N]$
Output: A set of intervals $I(Q)$

```
    I(Q) ← {}
    for i ← 1 to |Q| do
        curr_occ[i] ← −∞
    end for
 5: for rpos ← 1 to N do
        curr_occ[term_map(D[rpos])] ← rpos
        lsym ← argmin_{1≤i≤k}{curr_occ[i]}
        if curr_occ[lsym] > −∞ then
            I(Q)←I(Q) ∪⟨curr_occ[lsym]...rpos⟩
10:         curr_occ[lsym] ← −∞
        end if
    end for
    return I(Q)
```

Algorithm 2 provides the details. In this algorithm, the $D[\cdot]$ values in the direct file correspond to the term that is being processed. At each iteration the next term (or non-term) $D[rpos]$ is considered (step 6). A mapping to query terms (denoted *term_map*() in the pseudo-code) is used to convert raw term ID numbers across the whole collection to restricted local values pertinent to this query, with instances of the $|Q|$ query terms converted (uniquely) to values in $1\ldots|Q|$,

D_3: `Scalable Vector Graphics images can be produced by the use of a vector graphics editor.`

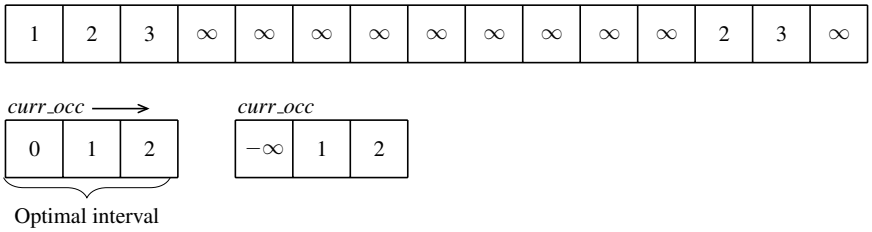


Figure 3.7: A running example of finding optimal intervals for query "`scalable vector graphics`" using the LPS approach. The grid in the middle represents a direct file of Figure 3.1, where all terms other than query terms are marked as ∞. The last row shows array *curr_occ* at position 3 and 4, respectively. The first position shows an example of optimal interval since all three values in *curr_occ* are filled. When position 4 is considered, the algorithm set the previous leftmost term "`scalable`" to $-\infty$.

and instances of non-query terms being converted to (say) $+\infty$. At each cycle, if inclusion of the symbol in $D[rpos]$ means that all of the $|Q|$ positions in *curr_occ* have positive values, then the right-hand end of an interval has been located. At this point, the interval is saved, and then the right-hand symbol of the interval is marked (Step 10), since it cannot take part in any future intervals.

Step 7 in Algorithm 2 makes use of an "argmin" operator over $|Q|$ values, and is presented this way in order to draw out the symmetry between the EPS and LPS implementations. Step 7 does not, however, require a heap and $O(\log k)$ time per execution. That cost can be avoided by maintaining an *lpos* index into D, and advancing it to the right (at step 7) when necessary, bypassing any non-query terms. Each time *lsym* is required, it can then be computed as *term_map*$(D[lpos])$. In terms of complexity, advancing *lpos* can require at most $O(|D|)$ time, and hence in an amortized sense, it is $O(1)$ time per loop iteration. The only other possible concern in Algorithm 2 is the cost of the term mapping *term_map*(); it can be implemented as a hash-table lookup with expected time $O(1)$ per call. Therefore, Algorithm 2 requires $O(|D|)$ time. Its correctness follows directly from the correctness of the eager PLANESWEEP approach [92]. If a frequency-based inverted file has been provided, where the TF value of each term is pre-computed, then LPS may terminate before scanning to the end of the document by adding an additional counter counting the occurrence of each query term being processed.

3.3.3 Comparing Input Structures

As has been noted, the two most common input representations are direct files and inverted indexes. Input via an inverted index is the "natural" arrangement for the eager PLANESWEEP process. The lazy mechanism offers more flexibility in that it can be coupled with both a direct file and an inverted file – all that is required is to merge the postings lists for the query terms, and construct an intermediate form we call a *condensed direct file*, as shown in Figure 3.8. For all query terms $q \in Q$, the cost of merging the postings is given by $O(T_{Q,D} \log |Q|)$, where $T_{Q,D} = \sum_{q \in Q} f_{q,D}$ is the sum of posting list lengths over the query terms. The cost of generating the condensed direct

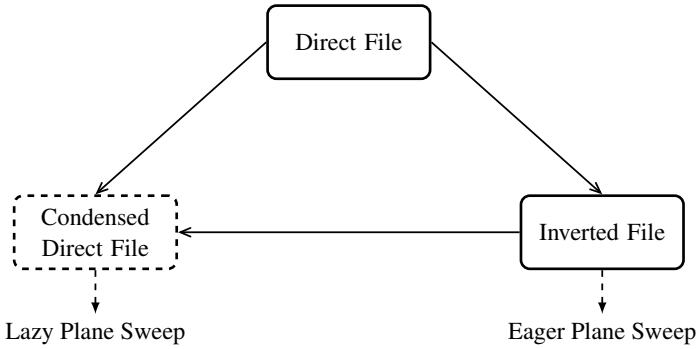


Figure 3.8: Input representations and methods

file is the same as the cost of running EPS (Algorithm 1) starting with an inverted file. Note that the condensed direct file is query specific, and for that reason cannot be pre-computed and stored.

Starting with a direct file, the specific condensed direct file for a query Q can be constructed in $O(|D|)$ time. Then, once the condensed direct file has been constructed, the EPS mechanism (Algorithm 2) can be executed in $O(T_{Q,D})$ time. Which of these various computation pathways should be chosen depends in no small part on the relative scale of $T_{Q,D}$, $|Q|$, and $|D|$. For example, a short query of rare terms against a long document will favor the use of an inverted file/EPS approach; whereas a long query that includes one or more frequent terms is likely to execute more quickly using the direct file/LPS approach.

Either way, in the case of proximity queries in the context of FDM and similar retrieval models, multiple related subqueries must be evaluated. This need brings further trade-offs, because it allows the cost of generating a condensed direct file to be amortized over the subqueries. This is one of the themes that is addressed in the next section.

3.3.4 Experiments on the Single Interval Finding Task

Experimental Setup. All experiments are performed on a machine using an Intel Xeon E5 CPU with 256 GB RAM running RHEL-v6.3 Linux, implemented in C++, and compiled using GCC 4.8.1 with –O2 optimization enabled. Execution times are measured in milliseconds (msec) per document. All experiments use the GOV2 document collection. Since all measurement focuses on the set of top-ranked documents retrieved for each query, the results are influenced primarily by the nature of the documents retrieved, and are agnostic to the overall collection composition. Each query in each set was then evaluated against GOV2 using Indri 4.7 with Krovetz stemming and a simple bag-of-words language model ranking, an out-of-the-box configuration, with stop words retained in the index. The 100 top-ranked documents for each query were retrieved, and used as a query-specific pool of 100 documents against which the interval-finding algorithms might be plausibly applied in an operational setting, and against which per document execution times could be measured.

The primary prerequisite to performing accurate comparisons of the algorithms we present is that a range of query lengths and document lengths be employed. To that end, two different query samples were created using the topics from the 2007–2008 Million Query Track (MQT)

Queries	Description
Set A	500 queries sampled from the 2007–2008 MQT logs, with exactly 100 queries for each length $k = 3$, 5, 7, 8, and 10
Set B	1,000 queries sampled from the 2007 MQT log, with a natural distribution of query lengths $3 \leq k \leq 12$

Table 3.5: Summary of the two sampled query sets generated from the TREC 2007–2008 Million Query Track topics. The document collection is GOV2 test collection, see collection statistics in Table 2.6.

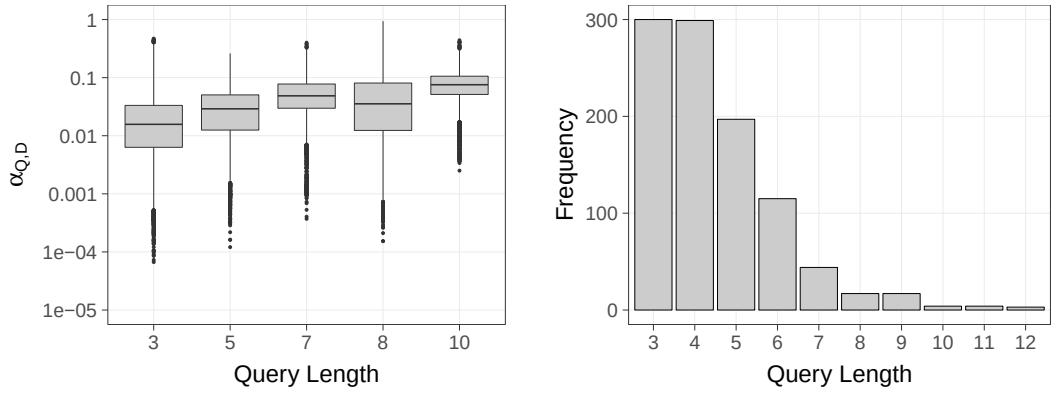


Figure 3.9: Statistics of the experimental datasets. The first figure shows term density ratio ($\alpha_{Q,D}$) as a function of query lengths, for all queries in Set A; the second figure shows the distribution of query length in Set B. The $\alpha_{Q,D}$ is computed using $T_{Q,D}/|D|$, where $T_{Q,D}$ is the frequency of all query terms in Q and $|D|$ is the document length.

tasks. Table 3.5 summarizes the composition of the two query sets. Set A was developed in order to compare performance characteristics as a function of query length, with a focus on longer queries, and a minimum of 100 queries of each length. The MQT did not contain 100 queries of length 9. We also generated a second query set containing 1,000 randomly selected queries with $3 \leq k \leq 12$, in order to capture the query distribution in a real query stream. Longer queries exist in the MQT log, but did not emerge as part of our sample. We refer to this second set as Set B, and use it to compare performance characteristics when the sample distribution is representative of a real query stream, with short queries more probable than longer ones. The distribution of lengths in Set B is shown in the left-hand side of Figure 3.9.

As discussed in Section 3.3.3, proximity evaluation depends on two key problem parameters: the length of the document, $|D|$; and the number of postings that must be processed, $T_{Q,D} = \sum_{q \in Q} f_{q,D}$. The *term density ratio* for a document D with respect to a query Q, is defined as $\alpha_{Q,D} = T_{Q,D}/|D|$, which captures the relationship between these quantities. The direct file-based approaches are likely to be most useful when $T_{Q,D}$ is high, and many of the terms in a document are also in the query. The $\alpha_{Q,D}$ distributions of the 50000 documents extracted from GOV2 using Set A queries are shown in the right-hand side of Figure 3.9. There is a slight trend for longer queries to be denser across the documents that they are to be applied to.

k	LPS-DF-S		EPS-IF-S		LPS-IF-S	
	med.	max.	med.	max.	med.	max.
3	0.019	0.611	0.003	2.335	0.002	2.304
5	0.033	0.765	0.004	2.263	0.004	3.147
7	0.055	1.224	0.008	5.428	0.013	5.869
8	0.066	1.054	0.011	9.350	0.020	9.626
10	0.085	1.139	0.014	6.628	0.029	7.778

Table 3.6: Median and maximum time (msec/doc) spent finding optimal $|Q|$-intervals using Set A queries.

Single-Query Interval Finding. We first compare different input structures with PLANESWEEP variants. As is shown in Figure 3.8, EPS is most amenable to an inverted file input (EPS-IF-S), and LPS to the direct file input (LPS-DF-S). But LPS can also be coupled with an inverted file input – a condensed direct file can be implicitly constructed from position lists without affecting the asymptotic cost of the interval-finding task, allowing an LPS-IF-S combination to also be tested.

Table 3.6 lists optimal k-interval finding for the Set A queries for these three methods, with the -S suffix used to differentiate between the interval enumeration problem. The two inverted file-based methods exhibit similar behavior, but with EPS-IF-S consistently a little faster than LPS-IF-S. Both inverted file-based methods are faster than LPS-DF-S for most queries. However, LPS-DF-S is better in the worst-case than both inverted file-based approaches, at all query lengths – situations in which $\alpha_{Q,D}$ is high. When this happens, locality of access in the direct file works better than iterating across several postings lists. Conversely, when $\alpha_{Q,D}$ is small, the skips induced from the postings lists outperform a sequential scan of the direct file for most query-document combinations.

Consider two queries, as a concrete example that follows. In the three word query: "`mental health counselors`", all three terms tend to have similar within-document term frequencies (around 200 per document), and all of the optimal 3-intervals appear close to each other. In this case, EPS-IF-S performs unnecessary comparisons and exchanges when trying to identify the rightmost position in the interval. However, in longer queries, such as "`what were the roles of african males in africa 1700s`" (recall that stop words are included) the method LPS-IF-S performs poorly (7.64 msec). The same query on the same document is much faster with LPS-DF-S, taking 1.10 msec, with $\alpha_{Q,D} = 0.16$.

Figure 3.10 captures these tensions, plotting the relative cost of constructing a condensed inverted file as a function of $\alpha_{Q,D}$ for Set A. As expected, when $\alpha_{Q,D}$ is small, it is much faster to start with an inverted file representation, whereas when $\alpha_{Q,D}$ is large, the condensed file format can be created more efficiently starting with the DF format. It seems conceivable that a query-sensitive mechanism might be devised to determine which input modality to use on a per-query basis if both formats were available at query time.

We now turn our attention to the performance of the interval-finding algorithms relative to $\alpha_{Q,D}$ and query length. Figure 3.11 plots interval-finding time as a function of $\alpha_{Q,D}$ for queries of length (a) $|Q| = 3$, (b) $|Q| = 5$, (c) $|Q| = 7$ and (d) $|Q| = 10$. The trend line for LPS-DF-

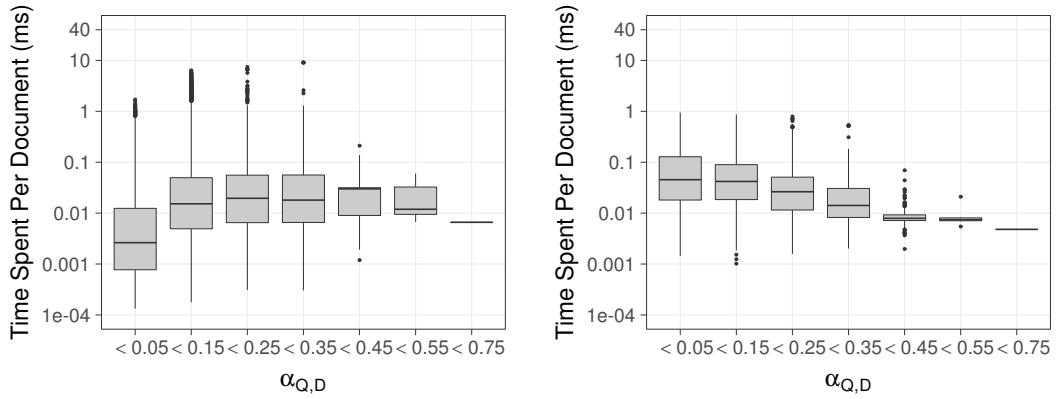
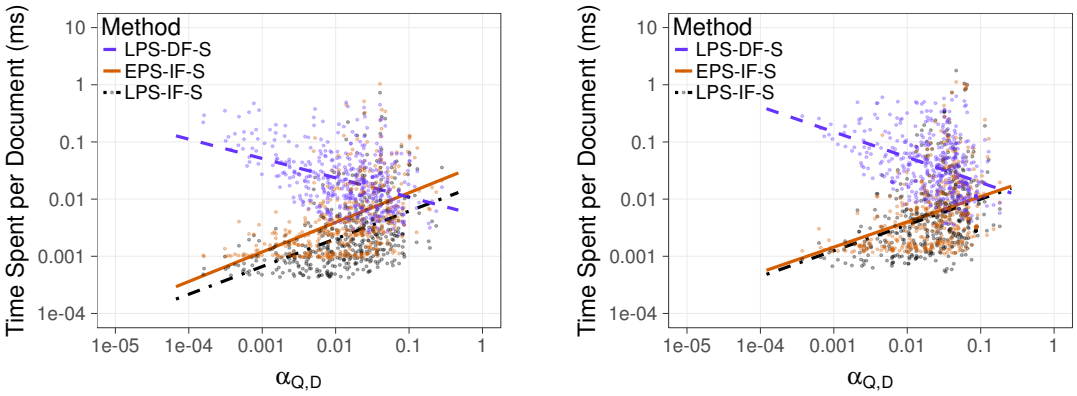


Figure 3.10: Time spent (msec/doc) constructing the condensed direct file, starting with an IF (the left-hand side) or a direct fie (the right-hand side), using the Set A documents. The y axis in both figures are plotted in a log scale. There were no queries in the $0.55 \leq \alpha_{Q,D} < 0.65$ range.



(a) 100 queries, $|Q| = 3$

(b) 100 queries, $|Q| = 5$

(c) 100 queries, $|Q| = 7$

(d) 100 queries, $|Q| = 10$

Figure 3.11: Time (msec/doc) spent as a function of $\alpha_{Q,D}$ for Set A. The two dashed regression lines represents LPS methods with colors differentiating two different input representations – the blue one represents a direct file as the input and the black one represent using an inverted file as the input. The red solid line represents EPS-IF-S method. The plotted points represent a sample of 5% of the total data in each graph.

S shows that query evaluation cost tends to decrease as $\alpha_{Q,D}$ increases; whereas for EPS-IF-S and LPS-IF-S the trend is for cost to increase. For example, the three term query "`effingham county ga`" with $\alpha_{Q,D} = 0.35$ performs poorly with both LPS-IF-S and EPS-IF-S, and is much more efficient with LPS-DF-S. The ten word query "`what was the population of the united states in 2000`" with $\alpha_{Q,D} = 0.19$ exhibits similar behavior. Comparing the four figures from (a) to (d), it can be seen that the "cloud" of points shifts to the right as the query length increases, confirming that the LPS-DF-S approach is more likely to be superior on longer queries.

In general, these experiments using single-query interval finding show that an inverted file with position offsets is a better choice than a direct file representation. However, the worst-case, while rare, is worth careful consideration.

3.4 Extracting Multiple Intervals

Up to this point, we have discussed methods for single interval extraction. However, as was mentioned in Section 3.1, the main cost of using the FDM model for retrieval is the task of optimal interval enumeration, which requires finding all optimal intervals for $2^{|Q|} - |Q| - 1$ subqueries. Let $\mathcal{I}$ be the set of all optimal intervals for all subqueries, where $\mathcal{I} = \{\mathcal{I}_2, \ldots, \mathcal{I}_{|Q|}\}$ and $\mathcal{I}_j$ is the set of optimal intervals for all length j subqueries. We now focus on the optimal interval enumeration problem, the larger task of computing interval statistics for a query Q and all of its subqueries Q'_i $(2 \leq i \leq |Q|)$.

3.4.1 Repeated Computations

One obvious solution to the problem of finding all optimal intervals is to simply repeatedly apply EPS (Algorithm 1) or LPS (Algorithm 2) multiple times, using each subquery as the input. However, the obvious pitfall is that the cost will grow exponentially as the query length increases, because of the growth in the number of the subqueries. It would be better if the running time for generating a solution was dominated by polynomial functions of $|Q|$ and N, rather than being exponential in $|Q|$.

Moreover, in an ideal situation the size of the computed output set $\mathcal{I}$ should also be a factor that bounds the cost of computation. That is, a useful algorithm is one that does the amount of work necessary to generate the required output, in this case $|\mathcal{I}|$, with an overhead cost as a function of N (or $T_{Q,D}$) and $|Q|$ that is additive rather than multiplicative.

One way in which time can be saved when multiple related subqueries Q' are being evaluated against the same document D has already been noted at the end of the previous section – the formation of a query-specific condensed direct file (CDF) which retains only (and exactly) the positions of the terms in Q that appear in D. The cost of constructing the CDF is the same as running a single-query interval finding mechanism: $O(T_{Q,D} \log |Q|)$ time if the input is an inverted file, or $O(N)$ time if the starting point is a direct file of length N, meaning that there is little point to the conversion operation for the single interval task. On the other hand, the conversion cost can be amortized over multiple executions for the all-subqueries task, and allows a more flexible range of

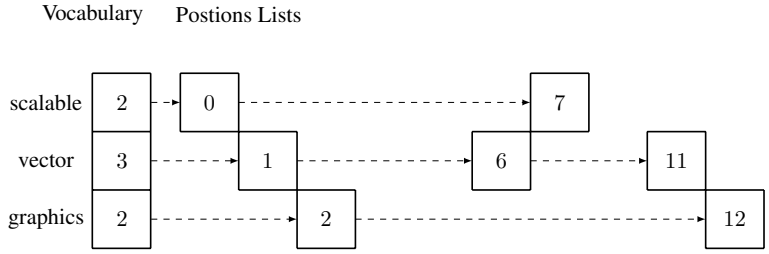


Figure 3.12: An example arrangement of the query "`scalable vector graphics`" using the inverted file. All positions of query terms are from the same document.

options. Hence, we can consider two relatively obvious baseline mechanisms for optimal interval enumeration, based on the two methods presented in the previous section:

- EPS-IF-S*: starting with an inverted file, apply Algorithm 1 a total of $2^{|Q|} - |Q| - 1$ times, reinitializing the postings list pointers and starting a fresh computation at each iteration.
- LPS-CDF-S*: starting with an inverted file or a direct file, construct a condensed direct file; then, using the CDF, apply Algorithm 2 a total of $2^{|Q|} - |Q| - 1$ times.

Both of these methods are used as reference points in the experiments that follow.

3.4.2 A Critical Observation

Consider an arrangement shown in Figure 3.12, with the example query "`scalable vector graphics`". We now focus on the query term "`vector`" at position 6 and consider its relationship to the other two query terms "`scalable`" and "`graphics`". There are no instances of "`scalable`" that appear between positions 1 and 6, the term "`vector`" at position 6 cannot be the right-hand end of an interval for "`scalable, vector`". On the other hand, the appearance of "`graphics`" between position 1 and 6 means that "`graphics`" at position 2 is the right-hand endpoint of a "`vector, graphics`" interval, spanning $[1..2]$. Similarly, consider position 7, which is the next possible right-hand point for intervals encountered during the PLANESWEEP process. The previous occurrence of "`scalable`" is at position 0. There is no possible benefit from considering "`vector`" at position 1 as a possible left-hand partner for "`scalable`" at position 7, because there is another instance of "`vector`" between position 0 and 7 that "shadows" it. Only the most recent instance of each symbol needs to be regarded as *interesting*; and "`vector`" at position 6 is an instance of "`vector`" that fits this definition, and defines the left-hand end of an "`scalable, vector`" interval. Position 2 is also interesting, because it is the rightmost "`graphics`" prior to the second "`scalable`", and hence can potentially be pinned on the left-hand end of intervals stretching through to "`scalable`" at position 7. Once the range $[2..7]$ is established as a viable one, all subsets of the set of distinct items strictly contained within that range need to be used to create intervals. Consider "`vector`" at position 6 as an example. There are two subsets to be considered $\{\}$ and $\{$"`vector`"$\}$, and thus two intervals $[2..7]$ and $[6..7]$ that span subqueries "`graphics scalable`", "`scalable vector graphics`" and "`vector scalable`", respectively. In total, three

intervals can be identified when focusing on position 7, which is the second occurrence of query term "scalable".

The next point to be focused on is the second occurrence of query "graphics" at position 12. But because the next position of the current left-hand term "vector" is 11, which is less than 12, there is no optimal candidate for query "scalable vector graphics". However, given that positions of "scalable" has been exhausted, interval $[6..7]$ is an optimal candidate for query "scalable vector". This observation shows that even when the next position of the current left-hand term is less than the position of the current right-hand term, it may still produce possible optimal candidate intervals. It also suggests that even if an optimal interval can not be formed for a query Q', however, it may be possible to have intervals for subqueries which are a proper subset of Q'.

3.4.3 A Lazy PLANESWEEP Approach for Finding All Intervals (LPS-IF-A)

More generally, each time an item *rsym* at position *rpos* becomes the right-hand focus of the sweep, all rightmost instances of other terms that lie between *rpos* and the prior occurrence of *rsym* can be used as the left-hand partners for intervals that terminate at *rpos*. The endpoints of that range define an interval over two terms; other intervals using the same range get added as a result of the need to include not just the endpoint items, but every combination of other terms that lies between them. Consider a running example in Figure 3.13 and assume current *rsym* is "scalable". According to the description, terms falling between position 0 and 7 can be recognized as candidates for the left-hand terms, which are "vector" and "graphics". Algorithm 3 captures these ideas. The main loop in Algorithm 3 iterates over potential interval right-hand positions in variable *rpos*, drawing them out of a merging process on the terms postings lists, as was the case in Algorithm 1. The term ID that matches this right-hand pivot value is stored as *rsym*. For any particular right-hand pivot point at *rpos*, there are at most $k - 1$ corresponding left-hand partners to be considered. Moreover, for a left-hand partner to be of interest, the most recent occurrence of the candidate must fall between *rpos* and the previous appearance of *rsym*. The location of the last occurrence of each symbol within the query is maintained throughout the computation in the array $curr\_occ[rsym]$. Initially all values in this array are initialized to $-\infty$. For example, position 6 will be kept for the term "vector" instead of 1 since it is the most "recent" occurrence. If such an intervening symbol *lsym* is found, then its position *lpos* forms an interval with *rpos*, and a two-element interval can be emitted. Once a two-element interval has been identified, every distinct item that occurs between *lpos* and *rpos* can also be (or not be) a member of an interval with the same bounding endpoints. To accommodate this requirement, the set of in-between items is accumulated in S, and then each subset $s \subseteq S$ is combined with the fixed endpoints *lpos* and *rpos* to create valid intervals. For example, when "graphics" is the left-hand point of an interval, an optimal interval for "scalable vector graphics" will also be emitted. The whole process completes when all occurrences of all terms are consumed from the postings lists. This occurs when every $next\_occ[i]$ value is $+\infty$. Note that the pseudo-code shows all output information being accumulated into one stream $\mathcal{I}$, but that separate streams can also be generated if required, one per occurring subquery.

Algorithm 3 Lazy Plane Sweep, All Intervals (LPS-IF-A)

Input: A query Q, and sorted position lists of all terms $\{P_1, P_2, \ldots, P_{|Q|}\}$
Output: All optimal intervals $\mathcal{I} = \{I(Q') \mid \text{for all } Q' \subseteq Q\}$

```
    I ← { }
    for i ← 1 to |Q| do
        curr_occ[i] ← −∞
        next_occ[i] ← FIRST(P_i)
 5: end for
    while (min_{1≤i≤|Q|} next_occ[i]) < ∞ do
        rsym ← arg min_{1≤i≤|Q|} {next_occ[i]}
        rpos ← next_occ[rsym]
        for lsym ←1 to |Q| do
10:         if curr_occ[lsym] > curr_occ[rsym] then
                // lpos can be LH end of intervals ending at rpos
                lpos ← curr_occ[lsym]
                S → {i | 1 ≤ i ≤ |Q| and lpos < curr_occ[i]}
                for all subsets s of S, including the empty set do
15:                 elems ← s ∪ {lsym, rsym}
                    I(elems) ← I(elems) ∪ [lpos...rpos]
                    I ← I ∪ I(elems)
                end for
            end if
20:     end for
        curr_occ[rsym] ← rpos
        next_occ[rsym] ← NEXT(P_rsym)
    end while
```

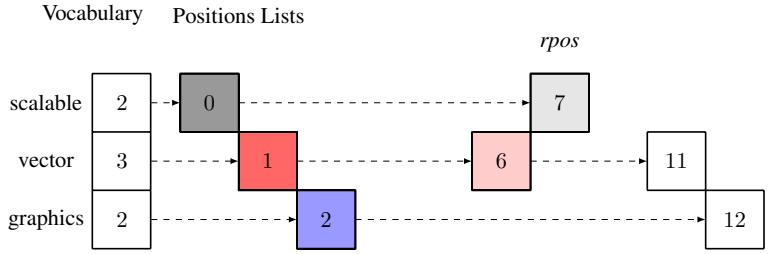


Figure 3.13: A running example of method LPS-IF-A when enumerating all optimal intervals of query "`scalable vector graphics`". The light gray block at position 7 is the current *rpos* of current *rsym* "`scalable`". The current *curr_occ* contains positions 2, 6, and 7. The box at position 1 that is marked as dark red will be dropped since the most recent position of the term "`vector`" is 6.

3.4.4 An Eager PLANESWEEP Approach for Finding All Intervals (EPS-CDF-A)

Algorithm 3 is a "lazy" implementation of the new approach. Algorithm 4 shows that the same idea can also be implemented in an eager manner using a condensed direct file. The primary loop iterates over potential left-hand positions via variable *lpos*. For each value of *lpos* a forward scan is made using *rpos*, and *rsym*, building intervals of increasing size until *lsym* = *rsym* and the bounding range for intervals whose left-hand end at *lpos* are found. The remainder of the process

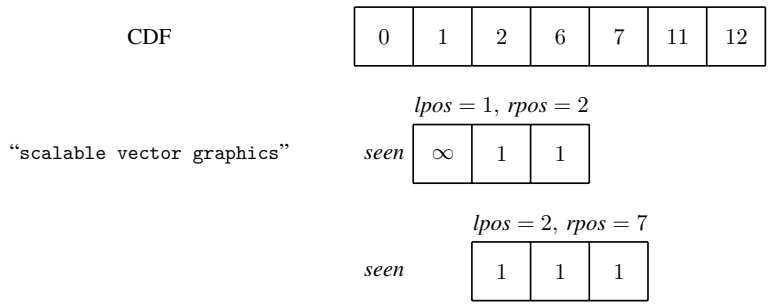


Figure 3.14: A running example of method EPS-DF-A when enumerating all optimal intervals of query "`scalable vector graphics`". The CDF is built using the inverted file from Figure 3.13. The *seen* array keeps current status of whether a term has been visited (represented using 1) or not (represented using ∞). The first *seen* array shows the status where $lpos = 1$ and $rpos = 7$, and the second *seen* array shows the status where $lpos = 2$ and $rpos = 7$.

is similar between the two implementations – within the range $D[lpos..rpos]$, all possible subsets of intervening items are used to form intervals. For example, consider our running example shown in Figure 3.14. When the left-hand position is fixed at 2 and then the algorithm continues scanning to the right, the first candidate $rpos = 2$ and *rsym* is "`graphics`", hence the last two blocks in *seen* are set to 1. However, when $rpos = 6$, *rsym* now is the same as *lsym*, which indicates the status where the left-hand term and position require an update. Therefore, when $rpos = 6$, the current $lpos = 2$ and the current *lsym* is "`graphics`".

One important difference is that in Algorithm 4 terms are considered in position order, and the use of bitvector *seen* allows set S to be implemented as a stack when the pseudo-code is translated into an actual program. On the other hand, in Algorithm 3, set S is rebuilt each time it is required by iterating over $|Q|$ possibilities. This factor can influence the relative execution times of the two methods in certain situations.

Finally, note that while Algorithm 4 is presented assuming that an inverted file data structure is used, it can also easily be adapted to a direct file.

3.4.5 Correctness and Analysis

This section first shows the correctness of the two proposed algorithms for finding all optimal intervals of all subqueries derived from a given query and then analyzing their cost.

Correctness. The discussion above already introduced the key notion that ensures that the two new algorithms are correct: *any optimal interval* $[lpos..rpos]$ *that ends with rsym cannot contain any other instances of rsym*. To see the truth of this claim, suppose a position v exists such that $lpos < v < rpos$ and (making use of the notation employed for the direct file) $D[v] = rsym$. Now consider the interval $[lpos..rpos - 1]$. Only one symbol, $D[rpos] = rsym$, has been excluded relative to the original interval $[lpos..rpos]$, and hence *rsym* is the only term that could possibly not also appear in $[lpos..rpos - 1]$. But there is another location v with $lpos < v \leq rpos - 1$ for which $D[v] = rsym$. Hence, $[lpos..rpos - 1]$ must be optimal for the same subset of terms as $[lpos..rpos]$, contradicting the assumption that $[lpos..rpos]$ is optimal. A similar argument

Algorithm 4 Eager Plane Sweep, All Intervals (EPS-CDF-A)

Input: A query Q, and sorted position lists of all terms $\{P_1, P_2, \ldots, P_{|Q|}\}$
Output: A set of intervals $\mathcal{I}$ for Q and all subqueries of Q

```
    (D', T_{Q,D}) ← BUILD_CDF({P_1, P_2, ..., P_{|Q|}})
    I ← { }
    for i ← 1 to |Q| do
        seen[i] ← 0
 5: end for
    for lpos ← 1 to T_{Q,D} − 1 do
        lsym ← D'[lpos]
        rpos ← lpos + 1 with lsym at left-hand end
        rsym ← D'[rpos]
10:     while rpos ≤ T_{Q,D} and lsym ≠ rsym do
            if seen[rsym] = 0 then
                // rpos can be RH end of intervals starting at lpos
                S ← {i | 1 ≤ i ≤ |Q| and seen[i] = 1}
                for all subsets s of S, including the empty set do
15:                 Q' ← s ∪ {lsym, rsym}
                    I(Q') ← I(Q')∪[lpos...rpos]
                    I ← I∪I(Q')
                end for
                seen[rsym] = 1
20:         end if
            rpos ← rpos + 1
            rsym ← D'[rpos]
        end while
        // reset seen to previous state
25:     for i ← lpos to rpos do
            seen[D'[i]] ← 0
        end for
    end for
```

shows that if $D[lpos] = lsym$, and $[lpos...rpos]$ is optimal, then there cannot be any occurrence of *lsym* in $D[lpos + 1...rpos]$.

In both Algorithms 3 and 4, all possible intervals that do not violate this requirement are identified, and all possible subsets of intervening items within such intervals are considered. Therefore both algorithms correctly generate all optimal intervals for the query Q and all of its subqueries $Q' \subseteq Q$ for which $|Q'| \geq 2$.

Analysis. We consider Algorithm 4 first. The main loop at step 6 iterates $T_{Q,D}$ times, where $T_{Q,D}$ is the size of the CDF, and is equal to the sum of the lengths of the term's postings lists. The inner loop at step 10 iterates a variable number of times, hunting forwards through D' for the next occurrence of *lsym*, the symbol at $D'[lpos]$. For each of the query terms, the loop at step 10 iterates at most $T_{Q,D}$ times, since each rightward scan starts at one occurrence of *lsym*, and stops when it reaches the next, forming a telescoping sum. The algorithm execute step 11 at most $k \cdot T_{Q,D}$ times over all values of *lpos*.

As already noted, S need not be re-evaluated each time step 13 is reached, because intervening symbols are added to it and never removed, allowing it to be stored compactly in a k-element array. Once S is formed, each iteration of the inner loop at step 14 takes $O(1)$ time, and adds one item to $\mathcal{I}$. Over all iterations of all loops, the total time spent in the loop commencing at step 14 is thus $O(|\mathcal{I}|)$.

If we have a query of length k, the only other cost is at step 1, and as already discussed requires $O(T_{Q,D} \log k)$ time if a heap is used to convert a set of postings lists to a CDF, or $O(N)$ time if the input is from a direct file. Note that in a pragmatic implementation, with k unlikely to be larger than around 10 or 15, an array (rather than a heap) is also a plausible choice for use during the inverted file to the CDF conversion process, taking $O(T_{Q,D}k)$ time.

Summing over all components, the execution time of Algorithm 4 is thus $O(T_{Q,D}k + |\mathcal{I}|)$ when the input is provided as an inverted file, and $O(T_{Q,D}k + |\mathcal{I}| + N)$ if the input is a direct file.

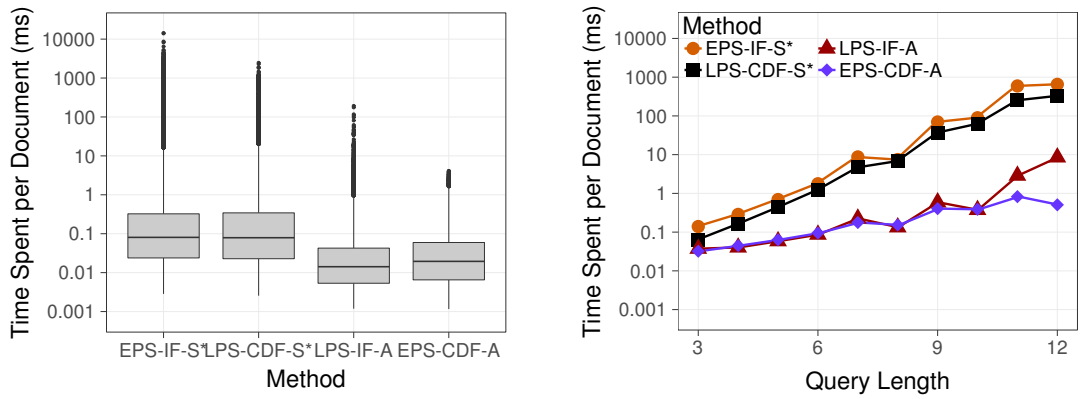
Algorithm 3 is harder to analyze. The loop at step 6 iterates $T_{Q,D}$ times, with $O(T_{Q,D}k)$ or $O(T_{Q,D} \log k)$ time spent merging postings pointers, depending on the data structure used. For each iteration the loop at step 9 executes exactly k times and some number – possibly as many as all – of those executions makes it past the test at step 10, to execute the block of statements starting at step 11. At step 11, the process of constructing S requires $O(k)$ time, because *curr_occ* is not ordered. Finally, step 14 requires $O(1)$ time per interval that is generated, as was also the case with Algorithm 4. Putting all of these parts together, it is possible that on some inputs Algorithm 3 requires as much as $O(T_{Q,D}k^2)$ time. That worst-case bound could be reduced by maintaining *curr_occ* in sorted order via a permutation vector, paying an overhead cost to rearrange it every time it changes. As is shown below, for typical inputs the extra expense of that rearrangement is not warranted in terms of average case performance. Although, not performing the rearrangement risks of more costly behavior on some queries.

3.4.6 Experiments on Finding All Optimal Intervals

As shown in Section 3.4, the problem of enumerating optimal k-intervals for all possible subqueries can be solved in a single-pass using either a Lazy or Eager evaluation strategy. In addition, there are a couple of other confounding factors. It is possible to create a condensed direct file representation once, and amortize the cost of the merge from an inverted file across multiple "single interval query" computations. In Table 3.7, two different approaches are considered when enumerating all of the possible $2^k - k - 1$ term patterns. The first approach is what systems currently do, that is, compute each interval separately. The LPS-IF-S* method runs Algorithm 1 multiple times, starting each time with the postings lists, with the -S* suffix intended to indicate that the single-query process is run multiple times. The LPS-CDF-S* method takes a different approach. It first creates a CDF by merging the position offsets from the inverted files. The CDF is then processed $2^k - k - 1$ times, each pass making use of Algorithm 2 to compute the intervals for one of the possible subqueries. The contrast between these two algorithms is captured by comparing the median execution time, and the worst-case execution time. For most query-document combinations EPS-IF-S* is the best choice. However, the additive worst-case is very costly, taking a full 3 seconds for one of the 10 word queries on a *single* document. The one-off cost of computing the

k	EPS-IF-S*		LPS-CDF-S*		LPS-IF-A		EPS-CDF-A	
	med.	max.	med.	max.	med.	max.	med.	max.
3	0.011	7.300	0.011	1.307	0.006	2.821	0.007	0.986
5	0.089	37.770	0.104	8.757	0.018	3.758	0.024	1.790
7	0.778	344.800	1.113	59.060	0.060	7.037	0.071	3.282
8	2.104	1303.000	3.102	142.600	0.098	18.640	0.108	5.016
10	9.315	3397.000	17.160	801.600	0.175	46.350	0.191	9.222

Table 3.7: Median and maximum time (msec/doc) to find the optimal k-intervals for all subqueries for each query. The experimental dataset is Set A listed in Table 3.9.



(a) Performance overview. (b) Performance breakdown by query length.

Figure 3.15: Time (msec/doc) to extract the optimal k-intervals for all subqueries using Set B (listed in Table 3.9). Figure (a) gives an overview of the per-query variation, incorporating all of the queries; figure (b) shows the average time cost for each method broken down by quey length. Both figures show the time cost of processing one document and are shown in log scale.

CDF dramatically decreases the worst-case performance, but it can still be inefficient in certain situations.

The key message from Table 3.7 is that performing the optimal k-interval enumeration process in a single-pass is dramatically more efficient than either of the two multiple-pass strategies, and, as expected, the gap becomes more pronounced as the query length increases. Although there are some small differences, the two -A PLANESWEEP approaches exhibit similar average performance when generating intervals for all subqueries. The key difference is again related to the worst-case. The tension between amortizing the creation of the CDF once and benefiting from locality of access, compared to the cost of hitting a rare long document with many occurrences of the terms, separates the two approaches in the worst-case.

Figure 3.15 shows the relative performance of multiple-pass and single-pass approaches. Note the use of a log scale. The upper panel shows the average performance for all 10,000 document-query combinations in Set B. Each boxplot summarizes the time values as follows: the solid horizontal line indicates the median; the box shows the 25th and 75th percentiles; and the whiskers show the range, up to a maximum of 1.5 times the interquartile range, with out-

liers beyond this shown as separate points. In the overall performance comparison for all Set B queries, the LPS-IF-A algorithm appears to be slightly better than EPS-CDF-A, but a t-test shows that the two are in fact significantly different with $p = 0.01$. This is largely due to the fact that the variance in query performance is more tightly bound for EPS-CDF-A than for LPS-IF-A. The bottom panel in Figure 3.15 shows the average (mean) processing cost in milliseconds per document as a function of query length. The two single-pass methods enjoy a near two-fold performance improvement for queries of length $|Q| = 3$, and a more than 1,000-fold improvement when $|Q| = 12$.

The experiments we have described in this section leave absolutely no room for ambiguity. The problem of optimal interval enumeration for all subqueries is greatly accelerated by our two new methods, even when queries have as few as $k = 3$ terms in them.

3.5 Conclusions

Using proximity features to improve retrieval effectiveness has received a great deal of attention within the literature, and a range of studies on comparing effectiveness gains when using different proximity-based models, for example term-pair models and higher-order proximity models have been conducted. However, the effectiveness of individual proximity features is unclear. Moreover, the computational cost of proximity features is high, and there is limited study that develops more general proximity feature extraction algorithms [92]. Although there are extensive studies on computing a specific type of proximity features, for example, phrase and term-pairs. If a proximity model requires statistics from multiple subqueries, then the computational cost grows exponentially in relative to the query length.

In this chapter, we have focused on the MRF retrieval model, which is currently considered a state-of-the-art proximity model. We analyzed the effectiveness gain of two main types of proximity features in MRF – phrases and unordered windows. Each type of proximity feature has been considered and evaluated individually, to observe how they affect the overall performance of the MRF model. The experimental study conducted in this chapter shows that, while phrase features can improve the MRF retrieval model to some extent, unordered window features tend to be more robust. Nevertheless, a combination of both can provide the greatest gain in retrieval effectiveness.

The high computational cost of extracting proximity features is considered in two steps: the process of extracting optimal intervals in a specific query, and the process of computing proximity statistics for many subqueries. First, we presented two different approaches for efficiently computing an optimal $|Q|$-interval based on input representation. We found that in most cases, starting with position offsets in an inverted file is more efficient than using a direct file. However, some query-document combinations can induce bad behavior in $|Q|$-interval computations. By exploring the efficiency of the algorithms as a function of the term density ratio, we were able to isolate exactly when each evaluation strategy and input format is preferable.

In order to improve the efficiency of models that leverage optimal intervals across many subqueries, we proposed, analyzed and evaluated two novel algorithms. By exploiting a key insight into the way intervals must be formed within subqueries, we showed it is possible to

extract optimal intervals for many subqueries in a single-pass. This crucial observation resulted new algorithms that significantly reduce the cost of the interval enumeration. The performance gap between current approaches and our approaches increases as query length increases, making it a viable tool in real-world search engines wishing to incorporate proximity computations into current ranking functions.

However, bottlenecks remain that may potentially lead to the inefficiency of higher-order proximity models. One cost is the use of global statistics in the retrieval model, especially for these higher-order proximities, which is unlikely to be accurately estimated, resulting in a two-pass evaluation over the inverted index. The trade-off problem between efficiency and effectiveness is considered in Chapter 4, where further improvements using higher-order proximity models is presented.

CHAPTER 4

Efficient and Effective Retrieval with Higher-Order Proximity Features

Bag-of-words models are widely used in information retrieval. Regardless of whether they are based on probabilistic ranking principles, such as BM25 [89], or drawn from a language modeling family, such as the Query Likelihood model [81], the models assume either that the query terms are fully or conditionally independent of each other. To compute a document score, each term contributes according to a TF statistic and IDF weighting, with the sum across the terms taken as the score for the document as a whole. Bag-of-words models are widely used because they can be efficiently implemented, and in most applications also attain good retrieval effectiveness.

As discussed in Section 2.2.3 and Chapter 3, proximity-based retrieval methods are one of the most widely used approaches among all other considered approaches that extend beyond bag-of-words models. Therefore, a range of sophisticated proximity models that allow terms to be dependent on each other have been proposed. These proximity-based retrieval models have shown a significant gain in retrieval effectiveness over bag-of-words models [24, 42, 45, 73, 78, 111]. The proximity model proposed by Metzler and Croft that is based on Markov Random Fields (MRF) has received considerable attention and it has been widely applied due to its high effectiveness. Improvements in effectiveness are achieved by considering all possible dependencies among query terms in the full dependency model (FDM); the trade-off cost arises in the form of higher computational costs. Bigram models can also be used as a compromise between full dependence, and fully independent models. However, the parameter tuning process for bigram models can be non-trivial, and may not obtain the expected effectiveness even if resources such as Google's N-grams are applied [8].

In order to achieve high effectiveness, current MRF term-dependency models require global collection statistics, counting the frequency of each possible dependency. Obtaining such global statistics requires traversing entire postings lists, including the positional index of each document, which is expensive. For sequential dependencies that consider relationships between two query terms, building a partial term-pair index may be viable and may reduce some computational overhead. However, unless large amounts of storage and preprocessing time are used, global statistics

describing two or more terms can only be computed at query time; the cost of doing that then becomes a significant bottleneck when evaluating term-dependency queries since each takes into account the scores as described in Chapter 3. In addition to accessing the positional information of the query terms in the collection, two complete iterations over the postings lists of terms must be performed for acquiring the global statistics. This means that long queries, or queries containing frequent words, may give rise to unacceptable query latencies.

There has been less attention paid to using alternative statistics for reducing the computational overhead. Macdonald and Ounis [66] were the first to examine trade-offs between effectiveness and efficiency from the perspective of global and local statistics, and concluded that using local statistics when computing proximity-based retrieval models is a reasonable trade-off. However, this conclusion was built on proximity features using term-pairs, without considering subqueries of length three or longer. When higher-order proximity features are considered in the retrieval models, such trade-offs are not clear. This chapter studies the relationship between effectiveness and efficiency when using only local higher-order proximity statistics in term-dependency models.

We begin this chapter by describing a number of proximity heuristics and different proximity-based retrieval models, reinforcing the importance of proximity features in the retrieval process, and confirming that it is crucial to combine both term-independent and term-dependent score components in order to achieve improved effectiveness. Then, to address the "global statistics" bottleneck associated with proximity models, we propose an alternative feature function that can be used in the MRF framework in Section 4.2. We empirically test our new variant MRF model on three TREC collections, and show its effectiveness compared to other MRF models in Section 4.3. Finally, having demonstrated the effectiveness of the approach, we turn our attention to execution cost, and show that our mechanisms reduce per-document retrieval costs by up to 60% compared to other methods.

4.1 Background

4.1.1 Proximity Heuristics and Features

We first describe the heuristics that form the basis of proximity features that can be applied within the context of existing unigram models. The different types of proximity features are also detailed, as one of our overarching goals is to derive proximity statistics that are not only local, but also allow a reasonable trade-off between effectiveness and efficiency as to their computationally costly global counterparts.

Influence of Proximity Scores. The question of whether proximity features boost bag-of-words scoring models arises because unigram models already incorporate some of the same influences. For example, if two query terms both appear in a document, then the document will almost certainly be scored more highly by a bag-of-words model than if only one of them appears, and it might be that the two occurrences are only rarely separated by a large span of terms anyway. In this case, using an auxiliary proximity model might not alter the underlying document ranking in

[0]*England* [1]*is* [2]*set* [3]*to* [4]*have* [5]*women* [6]*priests* [7]*within* [8]*two* [9]*years,* [10]*following* [11]*a* [12]*close* [13]*vote* [14]*in* [15]*the* [16]*Church* [17]*of* [18]*England's* [19]*general* [20]*synod* [21]*yesterday.* ... [50]*if* [51]*women* [52]*were* [53]*ordained.* [54]*Mr* [55]*Gummer* [56]*had* [57]*suggested* [58]*that* [59]*he* [60]*might* [61]*leave* [62]*the* [63]*Church* [64]*of* [65]*England* [66]*over* [67]*women's* [68]*ordination.*

Figure 4.1: A text section from a document that is relevant to the query "`women ordained church of england`" (Robust04, query 621).

any way. Part of our work here is to demonstrate the converse – that adding proximity features to the BM25 and LMDS unigram bag-of-words models unambiguously improves effectiveness, provided that the two parts are combined carefully. In particular, let $\mathit{Score}(Q, D)$ be the score of any bag-of-words model, $\mathit{Score}(\mathcal{I}, D)$ be the score derived solely from the model based on the proximity features in a document, where $\mathcal{I}$ is the set of all intervals derived from Q, and let $\Pr(D = 1)$ be the probability of a document being relevant. Then if proximity features are a useful addition to the bag-of-words model, we would hope to demonstrate that if $\mathit{Score}(Q, D_1) = \mathit{Score}(Q, D_2)$ and $\mathit{Score}(\mathcal{I}, D_1) > \mathit{Score}(\mathcal{I}, D_2)$, then $\Pr(D_1 = 1) > \Pr(D_2 = 1)$. We present results supporting this hypothesis in Section 4.3.3.

Proximity Feature Statistics. Section 2.2 briefly discussed different ways of categorizing proximity statistics. In this section, we focus on two types of proximity statistics: frequency-based and distance-based. Consider the example document in Figure 4.1 and the query "`women ordained church of england`". Proximity statistics are defined in a way similar to term frequency of unigrams, and counts the number of occurrences of an interval. Note that the definition of an *interval* is similar to Chapter 3, but may differ slightly in the optimality when considered in the context of specific models. Consider the interval definition in Chapter 3. In conjunction, we examine the set of intervals that is derived from the phrase "`Church of England`" from the sample document, where its frequency is two. Approaches for counting intervals within a specified distance were described in Chapter 3, and in the same way we can compute the distance of considered subqueries.

One of the key differences between the frequency-based and distance-based proximity statistics is the way in which these statistics are used as evidence for indicating the extent to what the current document is relevant to the given query. Frequency statistics are often applied directly, and the intuition is simple – the more frequently that an interval occurs within a document, then, the more likely it is that the document is relevant. A similar assumption is made by unigram models. From a methodological standpoint, how one defines a function capable of aggregating and translating the distance values within intervals into a reliable indicator of relevance is not obvious. There are various integrating methods, as shown in Table 4.1.

Roughly speaking, these methods fall into two categories: to consider interval distance per subquery [24, 42, 82, 111]; or to assemble distance values on a per unigram basis [17, 109]. The first three methods listed in Table 4.1 are straightforward. Consider the example document in Fig-

Method	Definition	Description
MinDist	$\min_{I \in I(Q')} \lvert I \rvert$	The MinDist takes all distance values as the input and uses the smallest distance value as the output [111].
MaxDist	$\max_{I \in I(Q')} \lvert I \rvert$	The MaxDist takes all distance values as the input and uses the longest distance value as the output [111].
AvgDist	$\frac{\sum_{I \in I(Q')} \lvert I \rvert}{\lvert I(Q') \rvert}$	The AvgDist computes the averaged distance over all intervals extracted for a subqery Q' [111].
Summation^1	$w_{Q'} \sum_{I \in I(Q')} \frac{w_I}{\lvert I \rvert}$	The Summation^1 method adds up all reciprocal distance values from the set of intervals capturing the same subquery Q' [24, 42, 82]. $w_{Q'}$ is a weight parameter indicating the importance of the current subquery and w_I is a parameter indicating the importance of the current interval.
Summation^2	$w_{Q'} \sum_{I \in I(Q')} \frac{w_I}{\sqrt{\lvert I \rvert}}$	A variant of Summation^1 [42] that uses the square root of the distance instead of the size of I.
Amortized	$\sum_{q \in Q'} \sum_{I \in I(Q')} \frac{\lvert Q' \rvert^\lambda}{\lvert I \rvert^\gamma}$	Interval distances aggregated over all unigrams in the subqueries [17, 109], using a simple summation method based on interval lengths. Both γ and λ are free parameters.

Table 4.1: Aggregation methods for distance-based proximity statistics, where Q' is one of the subqueries derived from the original query, $I(Q')$ is the set of intervals matching subquery Q', and I is a single interval.

ure 4.1 and the subquery "`england, women, church`". There are two non-overlapping intervals $[0...17]$ and $[18...64]$. MinDist will give 18 as the aggregate output, MaxDist gives 46 as the output and the AvgDist is 32. These three aggregation methods were explored by Tao and Zhai and they concluded that MinDist is highly correlated with the document relevance when compared to MaxDist and AvgDist. However, the studies were conducted using only term-pairs in the proximity features. Another widely applied integration method is summation, which also considers the subquery as a whole unit. Although two variants are shown in the table, they only differ slightly. The main idea of the summation approach is to sum the reciprocal distance values of all of the intervals matching a single subquery, and use this sum as one of the indicators to infer to what extent the current document is relevant to the query. For example, assume $w_{Q'} = w_I = 1$, then the method Summation^1 yields 0.07 and Summation^2 gives 0.38. All of the methods make use of the set of intervals by adding the total contribution to the matching subqueries. An alternative way of computing contributions of intervals matching a subquery is to consider each query term

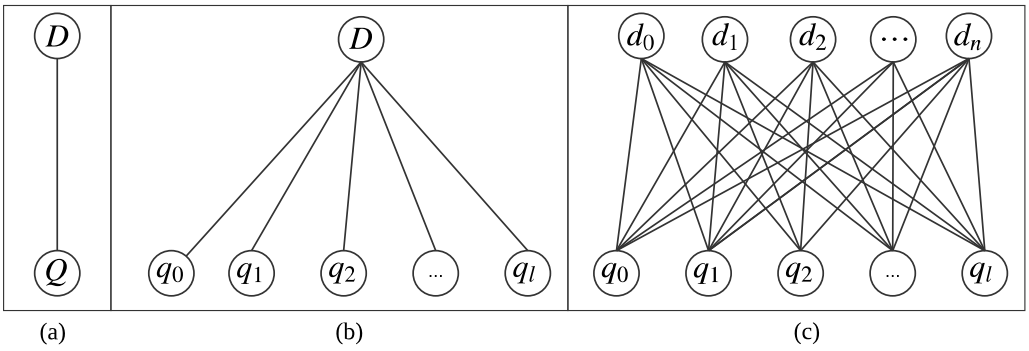


Figure 4.2: Three different dependencies modeled using MRFs, adapted from Metzler [72] (Figure 3.1). Pane (a) models the relevance between the query and a document by considering all query terms as a entire unit; pane (b) shows an alternative where the MRF models the joint distribution for all individual query terms; pane (c) shows the most sophisticated dependencies, where not only the query but also the document are considered at a term-level.

individually. In other words, instead of considering the contributions made by the set of intervals for the subquery "`england, women, church`", the aggregated distance values are added to each query term "`england`", "`women`" and "`church`", respectively. We use "Amortized" to refer this type of aggregation in Table 4.1. For example, consider the method proposed by Song et al. [109] and let $\lambda = 2$, $\gamma = 1$, which are the default values. Since the interval captures three query terms, the total contribution of the interval is $0.69 \times 3 = 2.08$. One advantage of the amortized computation is that the importance of an interval can be determined by the query terms it captures [17]. For example, when aggregating the interval distance values, Büttcher et al. use the IDF term weightings as part of the weighting scheme. However, their method considers only term-pairs. Both frequency-based and distance-based proximity statistics have been used in the literature, without conclusive evidence that one is better than the other. However, in all published work that reports the use of distance values, none claim to incorporate higher-order global statistics, and only term-level global statistics have been presented.

4.1.2 MRF-Based Proximity Models

We now introduce the background theory of MRF-based proximity retrieval models as proposed by Metzler and Croft [73]. Starting with the idea of using an MRF to model the relevance between a query and a document, this section then details the background on the different types of proximity relationships that exist and how these relationships can be represented using graph models.

Modeling Relevance using MRFs. A Markov Random Field is a probabilistic graphical model that is often used in statistical machine learning when describing joint distributions. It is represented as an undirected graph in which each node is a random variable, and each edge defines the dependence relationship. Figure 4.2 shows three possible ways of using MRFs to model relevance probability between a query and a document, as used by Metzler and Croft [73]. The three options

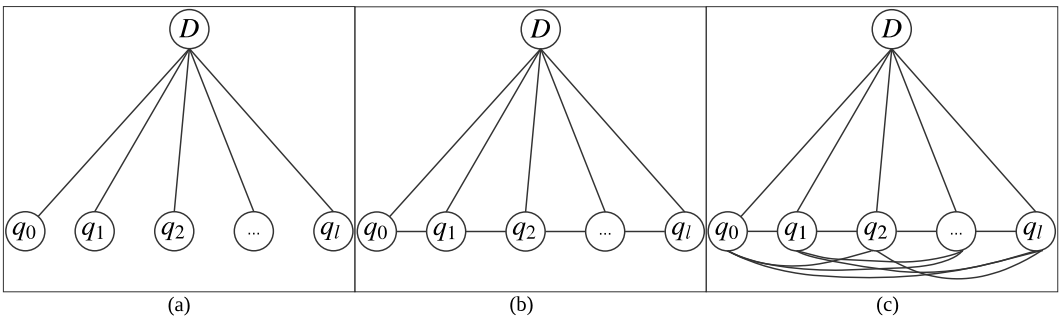


Figure 4.3: Three different dependencies in graphical representation, adapted from Metzler and Croft [73]. Pane (a) plots the full independence (FI), where there is no relationship between each query term; pane (b) plots sequential dependency (SD), where the relationships in the query terms are only between terms that are consecutive; pane (c) shows full dependency (FD), which assumes all query terms are dependent upon one another.

are shown in increasing complexity. The first model considers the query as a whole, regardless of the query length, where the query and a document are plotted as two nodes, with an edge indicating the potential relevance probability between the two. The second form of dependency considers query terms instead of the entire query. Hence the relationships are made between each individual query term and the document, so each query term is a node in the graph, as shown in pane (b) of Figure 4.2. The most complex relationship is shown in pane (c) of Figure 4.2, where not only the query but also the document is considered at a term-level. For the same document D, the first two models consider it as a whole unit, but the model in pane (c) splits it into d_0, d_1, ..., d_n nodes, and then connects them to each term in the query. Metzler and Croft use the second option to model the probability distribution of relevance between a query and a document, since the first model is too coarse to capture the relevance relationship, and the last one is too complex in practice.

Modeling Dependencies using MRFs. As shown in Figure 4.3, there are three types of dependencies among query terms, already briefly discussed in Section 2.2. The full independence model (FI) model assumes no dependencies between query terms and all bag-of-words retrieval models satisfy this assumption, with no proximity features considered. A more advanced model considers the sequential dependencies among query terms, where the phrase proximity features are considered. In most applications, only bigrams are considered in sequential dependencies. The model plotted in the third pane assumes a full dependency relationship, where query terms are dependent on each other, in any order. In the last type of model not only are phrase features considered, but also unordered windows.

We consider FDM as an example for explaining the modeling process. In total, there are three types of cliques in FDM: (i) cliques $\{D, q\}$ $(q \in Q)$, which represent relationships between each term and the document; (ii) cliques $\{D, q_i, q_{i+1}, q_{i+2}, \ldots\}$, which consider relationships between consecutive query terms and the document, and there is no limit on the number of query terms; and (iii) cliques $\{D, q_i, q_j\}$ $(i \neq j)$, where the relationships are considered among all query terms,

without order constraints, and where the cliques may contain more than two query terms. One key component in building the MRF is to define potential functions for the maximal cliques. This is done by first considering feature functions for each clique. As pointed out by Metzler [72], both language model and BM25 ranking functions can be applied as a potential function over the set of cliques. The widely applied MRF-based models implemented in Indri adopted the LMDS approach (Equation 2.5). Consider a subquery Q' and the set of intervals found in the document D for this subquery, then the feature function for clique $\{D, Q'\}$ is computed as:

$$\mathit{Score}(I(Q'), D) = \log\left(\frac{\mu \cdot f_{I(Q'),\mathcal{C}}/|\mathcal{C}| + f_{I(Q'),D}}{\mu + |D|}\right),$$

where $f_{I(Q'),\mathcal{C}}$ is the number of matching intervals in the whole collection, and $f_{I(Q'),D}$ is the within-document frequency of the intervals. This computation process is repeated for all intervals and all subqueries, using the LMDS scoring regime as a feature function. Once all components have been computed, they are combined to the final score of the document. Although a limited window size is applied in the scoring process, for simplicity this form of FDM omits the differences between cliques in the same clique set. Since only the frequency-based statistics are employed in the model, intervals in the same clique set are regarded as being the same, without further discrimination.

The MRF-based proximity models are still considered to be one of the best known higher-order proximity models, and the experimental results of Huston and Croft [45] on the Robust04 and on TeraByte tracks show that significant gains are possible compared to lower-order computations. Whilst effective, this model is expensive to compute for two reasons: (i) a large number of subqueries must be enumerated (exponential in $|Q|$); and (ii) the need for global proximity statistics mandates a two-pass retrieval process. Macdonald et al. [67] observe that one way to avoid computing the global statistics is to use a constant, as is done in the Ivory System[1]. However, it is unclear how best to tune this value, and it may be sensitive to the collection being used.

4.1.3 Possible Trade-Offs Between Efficiency and Effectiveness

Although MRF-based proximity models are effective, their computational cost is high. One reason these models have a high computational cost is the use of global statistics in the model, that is, the $f_{I(Q'),\mathcal{C}}$ component. We consider several options when computing MRF-based FDM models. The first is the choice of proximity statistics. Section 4.1.1 discussed both distance-based and frequency-based statistics and we consider whether it is possible to use distance values instead of frequencies in the retrieval models. Our hypothesis is that by carefully aggregating distance values, it is possible to use local statistics and reduce the computational cost of collection-wide statistics for higher-order proximity features. The second option we consider is the feature function and the potential functions. There is no rule of thumb as to which potential function should be applied when considering the distance heuristic. Nevertheless, most models that leverage distance values were BM25 formulations, which motivates us to investigate the difference between using BM25 and LMDS as potential functions.

[1] http://lintool.github.io/Ivory/index.html

4.2 Using Local Proximity Statistics in Retrieval Models

We now describe an alternative higher-order proximity retrieval model that is derived from MRFs. As previously discussed, there are two alternatives when designing the MRF-based proximity retrieval model: (i) by using distance-based proximity statistics and; (ii) through the use of the BM25 potential function instead of LMDS. This section focuses on this problem and mainly considers the design of a retrieval model with local proximity statistics only, starting from the smallest building blocks – intervals, detailed in Section 4.2.1. The scoring method is covered in Section 4.2.2. Then we consider how to aggregate intervals for all subqueries in Section 4.2.3 and show possible variants of our proposed models in Section 4.2.4.

4.2.1 Intervals

We follow the interval definitions in Chapter 3, which were originally proposed by Clarke et al. [24]. Among all interval variants, non-overlapping intervals are considered in this work, as suggested by Song et al. [109]. That is, the next candidate interval of the current subquery must start at a position beyond the right end of the previous optimal interval. For example, consider a subquery $Q'_0 = \{$"`women, england`"$\}$ and the example text in Figure 4.1. The optimal intervals are:

$$I(Q'_0)' = \{[0...5],\ [5...18]\ [18...51],\ [51...65],\ [65...67]\}$$

Since only the non-overlapping intervals are considered in the modeling process, the candidate set is:

$$I(Q'_0) = \{[0...5],\ [18...51],\ [65...67]\}$$

Similarly, consider the three term subquery $Q'_1 =\{$"`women ordained england`"$\}$, the corresponding set of subqueries are:

$$I(Q'_1) = \{[18...53],\ [65...68]\}$$

Note that for one subquery, we considered non-overlapping intervals, but intervals of different subqueries may overlap. For example, interval $[65...68]$ in $I(Q'_1)$ overlaps with $[65...67]$ in set $I(Q'_0)$. The smaller interval is not dropped, since it arises from a different subquery.

We consider proximity features by modeling full dependencies among query terms, as shown in Figure 4.3 (c). Each interval represents a clique, for example $[0...5]$ in $I(Q'_0)$ represents the clique $\{D$, "`women`","`england`"$\}$ and $[18...53]$ in $I(Q'_1)$ is a clique $\{D$, "`women`", "`ordained`", "`england`"$\}$. Note that in this process, all non-overlapping interval are kept, without considering distance constraints. The interval-finding process is iterated until the positions for all query terms are exhausted, and can be accomplished using the method described in Section 3.4.

4.2.2 Scoring Intervals

Since each extracted interval represents a clique, it is then scored using a feature function. The simplest method is to assign a score of zero or one, indicating only the presence or absence of the interval. Whilst the frequency-counting approach is simple and easy to understand, the main

drawback is that it can barely differentiate between two intervals. If two intervals capture the same subquery, using a counting method ignores the distance difference which is an important indicator in proximity heuristics. As one of our purposes is to explore the effect of using distance as statistical proximity feature, we define the function in such a way that an interval will be scored more highly if it:

- captures more query terms over a shorter distance;
- captures more important terms; and
- is not bounded by common terms.

The first of these relationships is easiest to motivate, and reflects the proximity hypothesis [42]:

> "The closer together a set of intersecting terms the more likely they are to indicate relevance."

This relationship has been applied in many proximity-based retrieval models. For example, in MRF models, only windows of up to some maximum length are considered, and the similar idea has been applied to other frequency-based models. Distance-based methods often directly capture the closeness of terms by giving different weightings according to both interval length and the subquery length, which indicates the sparsity of query terms distributed in the current interval, for example, the mixed distance-based method proposed by de Kretser and Moffat [34].

The second relationship is used to distinguish different subqueries. Although all types of cliques are considered in the modeling process in order to better capture term-dependencies, they may be given different weights allowing the specification of differing levels of importance for each type of clique within the model. There are two different approaches to implement this effect: using IDF weighting in the current collection, or a complex weighing that is trained with external resources, such as the weighting scheme adopted in Weighted Sequential Dependency Models (WSDM) [9]. We prefer the more straightforward option, which uses the IDF weightings of the terms. Our focus is on the local proximity statistics motivated by Büttcher et al. [17], therefore we consider the IDF of unigrams instead of each subquery in order to avoid a costly two-pass query evaluation process.

The third relationship further distinguishes between intervals that match within the same subquery. Assume there are two optimal intervals that capture the same subquery and both span the same distance. The heuristics considered so far may give them the same score. In this case, we use the boundary term to determine whether the current matching is an imprecise representation of the current subquery. If one of the intervals is bounded by common terms, such as stop words, it may be less informative when compared the alternative one. As a concrete example, consider the interval statistics for another subquery: $I(\text{“}\texttt{woman, of, england}\text{”}) = \{[0 \ldots 17], [18 \ldots 64]\}$. Both intervals are right bounded by the term "of", which is a common term. Although both of these intervals are valid candidates for this subquery, if the document also contained a valid interval such as "`woman of england`" that had the "of" in the middle, then all other things being equal, that third one should be favored. Indeed, intervals in $I(\text{“}\texttt{woman, of, england}\text{”})$ have much the same effect as intervals in $I(\text{“}\texttt{woman, england}\text{”})$ when "of" is one of the two endpoints.

To embody these three relationships, we propose that the candidate interval $[p_\ell \dots p_r]$ of a subquery Q' be scored as:

$$\begin{aligned} \textit{Score}([p_\ell \dots p_r], D) &= \frac{w_\ell^{(2)} \cdot w_r^{(2)}}{\bar{d}^2} \\ \bar{d} &= \frac{p_r - p_\ell + 1}{\sum_{q \in Q'} \min\{w_q^{(2)}, 1.0\}} \end{aligned} \tag{4.1}$$

where $w_\ell^{(2)}$ and $w_r^{(2)}$ are the IDF weights of the left-hand and right-hand term of the interval respectively, and $p_r - p_\ell + 1$ is the interval length computed using the current position of *lsym* and *rsym* boundaries. The $w_r \cdot w_\ell$ component discounts intervals that are bounded by common terms (the third relationship), and $\bar{d}$ can be viewed as a weighted average distance of the current matching interval. Computing the interval length using IDF values follows from the second relationship, where the importance of an interval capturing many common terms will contribute less when compared to an interval containing terms with a higher IDF. We borrow the distance calculation from Büttcher et al. [17], which is an empirical choice among all possible calculation options, and use $(p_r - p_\ell + 1)^{-2}$, which is a simple variation of the method proposed by Song et al.. In addition, when computing $\bar{d}$, rather than count the number of query terms in the subquery, we use $\min\{w_q, 1.0\}$ to down-weight subqueries that consist of low weighted terms, thereby also incorporating the second relationship. For example, if the weightings for the terms in the example query "`women ordained church of england`" are 1.1, 2.0, 2.3, 0.1, and 1.5, respectively, then the score of the interval $[0 \dots 5]$ in the example document is 0.18, and the score of $[0 \dots 17]$ is 0.002. This factor is computed automatically during query processing, and does not require additional parameter tuning.

4.2.3 Combining Interval Scores

Trotman et al. [113] observe that there is no single ranking function that is consistently better than the others, therefore, either LMDS or BM25 can give reasonable performance of the full independent (or bag-of-words) component. Although both a BM25 and a LMDS ranking method can be applied as the potential function in MRFs, we prefer BM25 since most of the distance-based statistics were used in conjunction with a BM25 method. Let $\textit{Score}(Q', D)$ be the score of set of intervals $I(Q')$ derived from a subquery Q'. For our purposes the score for these intervals is defined in a BM25-like way as:

$$\textit{Score}(Q', D) = \frac{\sum_{[p_\ell \dots p_r] \in I(Q')} \textit{Score}([p_\ell \dots p_r], D) \cdot (1 + k_1)}{\sum_{[p_\ell \dots p_r] \in I(Q')} \textit{Score}([p_\ell \dots p_r], D) + K} \tag{4.2}$$

where K is the BM25 parameter (see Equation 2.3),

$$K = k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\textit{avg\_}|D|}\right);$$

and where $[p_\ell \dots p_r]$ represents an interval found in the document. For each subquery, Equation 4.2 scores all non-overlapping optimal intervals enumerated using Algorithm 3 or 4. As noted

earlier, one of the reasons that FDM is expensive is because it uses collection-based frequencies, and hence requires two-pass through the index. To avoid that cost, we refrain from any use of global higher-order counts in Equation 4.2. As a surrogate, term-level IDF statistics were used in order to achieve better effectiveness, a design choice that has also been leveraged by many other researchers [17, 82, 109]. We also include ordered phrase relationships in our overall formulation, as we concluded in Section 3.2 that using both ordered and unordered proximity feature improves the retrieval effectiveness the most.

Let $\mathcal{Q}$ be the subquery set, and $\mathcal{Q}_0 \subset \mathcal{Q}$ be the set of sequentially dependent subqueries. Then each subquery in $Q' \in \mathcal{Q}_0$ captures the relationship in Figure 4.3(b), that is, each subquery in this set contains adjacent query terms $Q' = \{q_i, q_{i+1}, q_{i+2}, \ldots\}\,(1 \leq i < |Q|)$. We combine all of these features in the usual weighted manner, capturing dependencies in Figure 4.3(c) and using the BM25 derived potentials in MRF, to obtain our proposed method:

$$\begin{aligned} \mathit{Score}(Q, D) = (1-\lambda) \cdot \sum_{q \in Q} \mathit{Score}(q, D) \\ + \lambda \cdot \left(\sum_{Q' \in \mathcal{Q}_0} \mathit{Score}(Q', D) + \sum_{Q' \in \mathcal{Q}} \mathit{Score}(Q', D) \right), \end{aligned} \tag{4.3}$$

that is, $(1-\lambda)$ times the FI score, plus λ times the combined phrase score (over $Q' \in \mathcal{Q}_0$) plus the proximity score (also over $Q' \in \mathcal{Q}$). As BM25 is used as a potential function, the FI component is computed using Equation 2.3. The other two components are computed using Equation 4.2.

Equation 4.3 makes it clear that Local k-term Proximity (Lkp) – the name we give to this approach – expresses both independent relationships as well as sequential and full dependencies among query terms, both of which may improve query effectiveness; and in doing so, echoes factors captured by FDM. Since this model attempts to include all possible dependencies, all subqueries derived from the original query are considered, and thus the cardinality of subqueries grow rapidly relative to the query length, leading to significant costs. However, global weightings are only required for unigram terms, which makes this model more amenable to further efficiency optimizations. Moreover, our model benefits from optimal intervals, and, as described previously, does not require that a distance constraint be added as a further parameter to be tuned.

4.2.4 Variants

There are several alternative design options that can be made under the general ranking scheme. The first is the interval length, which is referred to as "window size" in some proximity models. We only consider optimal intervals, which form naturally according to the arrangement of terms in the document. Restricting interval lengths will filter long intervals, with little impact on efficiency. Moreover, since our model is a variant of the original MRF model, we can select clique types to be considered, thus if only bigram subqueries are considered, it reduces to a bigram model. We consider three versions:

- Lkp: as described by Equations 4.1, 4.2, and 4.3, which considers all dependencies, which scores a document based on all optimal intervals found for all subqueries derived from the

Collection	N	$\lvert\mathcal{C}\rvert$	Queries	
			T-query	U-query
TREC-8	0.5 M	253.4 M	250	1,933
GOV2	25.2 M	23,451.8 M	150	1,496
ClueWeb09B	50.1 M	40,416.4 M	3×50	–

Table 4.2: Test collections used, where "# Docs" is the number of documents, and $|\mathcal{C}|$ is the total number of terms. The two "Queries" columns list the number of queries for each collection, with "T" denoting TREC title queries, and "U" denoting user queries [6].

original query Q. Note that this model uses intervals without distance and order constraints. Also, as indicated by the name, higher-order proximity features are used.

- L$k$pf: as for Lkp, but with intervals restricted to a maximum distance, set as $4 \cdot |Q'|$, as the configuration suggested by Metzler and Croft [73]; and
- L2p: as for Lkp, but considering bigrams only, that is, the subquery set $\mathcal{Q}$ only contains subqueries that: (i) consists of term-pairs and; (ii) each term-pair is order preserving. There is no distance restriction in this model, unlike L$k$pf.

4.3 Experiments

We now explore the effectiveness and efficiency trade-offs of Lkp and its variants, starting by describing the datasets and test collections in Section 4.3.1. Before investigating the proposed models, we consider the impact of the proximity features and compare them with the full independence models in Section 4.3.3. Section 4.3.4 shows the effectiveness when using the proposed models and variants compared to existing solutions. Finally, we provide some exploration on the effectiveness and efficiency trade-offs in Section 4.3.5.

4.3.1 Experimental Materials and Evaluation Metrics

We use the Robust04, TB04–06, and ClueWeb09-Category-B (CW09B-2010 to 2012) test collections in these experiments, always in non-stopped forms, since proximity and phrase-based models can use combinations of stop words to improve overall effectiveness for some queries (consider the query "`the who`" for example). Indexing is carried out using Indri, with a Krovetz stemmer. Two broad types of query sets are used. The primary resources are the TREC title queries of the Robust04 task (Rob04, with 250 queries in total), the TeraByte Task from 2004 to 2006 (TB04–06, with 150 queries in total), and the three Web Ad Hoc Tasks from 2010 to 2012 (CW09B-2010 to CW09B-2012, with 50 queries each). As a secondary resource, we augment these with the user-generated query variants collected by Bailey et al. [6] in connection with the Robust03 and TeraByte04 Tasks, denoted here as Rob03-U and TB04-U. Table 4.2 summarizes the three document collections and seven query sets used; and Figure 4.4 summarizes the lengths of the five TREC title query sets.

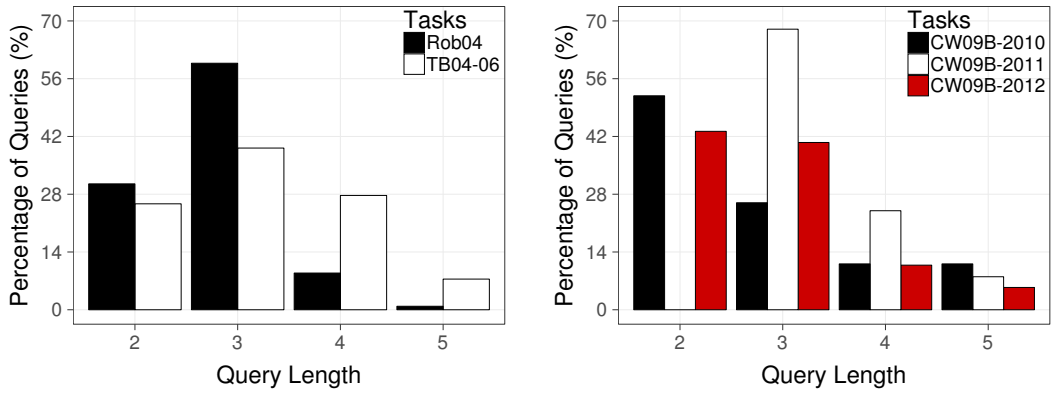


Figure 4.4: TREC title query lengths for five query sets: Robust04 (a subset of Topics 301–450, 601–700), TB04–06 (Topics 701–850), CW09A-2010 (Topics 51–100), CW09A-2011 (Topics 101–150), and CW09A-2012 (Topics 151–200).

The top $1{,}000$ documents are identified for each query in all effectiveness measurements. We choose our evaluation metrics and evaluation depth based on the pooling depth, thus the evaluation configurations are across collections. Note that different evaluation configurations may lead to different conclusions on the system effectiveness and we discuss this impact further in Chapter 5. The Robust04 and GOV2 results are reported using two relatively deep metrics, Average Precision (AP) and Rank-Biased Precision (RBP(0.95)), and in both cases are based on the full run and all available judgments. The three ClueWeb collections have shallow judgments, and hence are scored using expected reciprocal rank to depth 20 (ERR@20) and a more top-weighted rank-biased precision (RBP@0.8), again based on the full run and all available judgments. Of these three metrics, ERR@20 is computed using graded relevance, with the maximum gain set according to the corresponding task; while RBP and AP are both computed using binarized judgments. Experiments were performed on an Intel Xeon E5 CPU, 256 GB RAM, and RHEL-v6.3 Linux, implemented in C++, and compiled using GCC 4.8.1 with –O2 optimization.

4.3.2 Retrieval Models and Configurations

The TREC title queries are used as effectiveness baselines. We use BM25$^{(2)}$ with Robertson-Walker $w^{(2)}$ IDF weighting as the baseline FI model, and also list the effectiveness scores for Indri's built-in version of Okapi BM25$^{(1)}$, which uses Robertson-Spärck Jones IDF weighting $w^{(1)}$. Methods from the language modeling family are also included; these results are generated using Indri and the default configurations. To validate the experiments of Huston and Croft [45], we also include the BCTP bigram model as a baseline. Parameters for these reference systems are listed in Table 4.3.

The Lkp method described in Equations 4.1, 4.2, and 4.3 is included in Table 4.3, as are the variant, L$k$pf, that employs a distance constraint (set to match the MRF models, that is, $4 \cdot |Q'|$), and the L2p method, which makes use of bigrams only. Parameters for the three new methods were trained using the Robust04 queries on TREC-8 collection, and then applied to the others.

Models	Type	Global lvl.	Params.
BM25$^{(1)}$	BM25	term	$k_1 = 0.9$, $b = 0.4$
BM25$^{(2)}$	BM25	term	$k_1 = 0.9$, $b = 0.3$
LMDS	LM	term	$\mu = 2500$
SDM	LM-B	bigram	$\lambda_o = 0.15$, $\lambda_u = 0.05$
FDM	LM-N	n-gram	$\lambda_o = 0.1$, $\lambda_u = 0.1$
BCTP	BM25-B	term	–
L2p	BM25-B	term	$\lambda = 0.4$
Lkp	BM25-N	term	$\lambda = 0.4$
L$k$pf	BM25-N	term	$\lambda = 0.4$

Table 4.3: Retrieval models used in experiments. "Type" indicates the family origin of the method, with suffix "B" denoting a bigram model, and "N" a higher-order proximity model. Column "Global lvl" indicates the highest level that global statistics are used; and "Params" lists the parameters, with bag-of-words parameters re-used in the proximity models and not listed a second time. For FDM and SDM we use the recommended configurations in the original paper, and for BM25 we use the settings recommended by Trotman et al. [113]. The parameters for the three new models were trained using the Robust04 test collection.

4.3.3 Effectiveness of Proximity Features

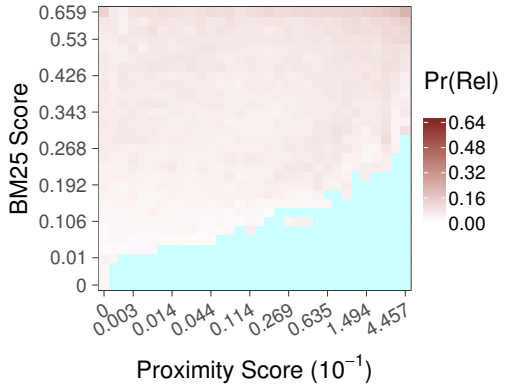
Our first experiment tests the hypothesis stated in Section 4.1.1: that the additional proximity features included in Lkp and L$k$pf are not captured by the simpler bag-of-words models. Taking *P-Score*(Q, D) as the measured proximity score, which consists of both ordered and unordered intervals. *FI-Score*(Q, D) as the score of any bag-of-words model, we wish to know, for two documents D_0 and D_1 that have (approximately) equal scores according to *FI-Score*$(\cdot)$, whether there is a residual effect that reveals a correlation between their *P-Score*$(\cdot)$ scores and the probability of being relevant.

We use the TREC title queries with more than one query term, together with Rob03-U and TB04-U in this experiment. Queries are treated as being independent even if derived from the same topic. On average, there are $1{,}512$ judged documents per topic for Rob03, of which an average of 78.15 are relevant. That is, across the judged documents the average probability of relevance is 5.04%. For the GOV2 TB04 task there are an average of $1{,}185$ judged documents per topic and 216.7 relevant, giving a background relevance probability of 18.73%. The experimental process carried out was:

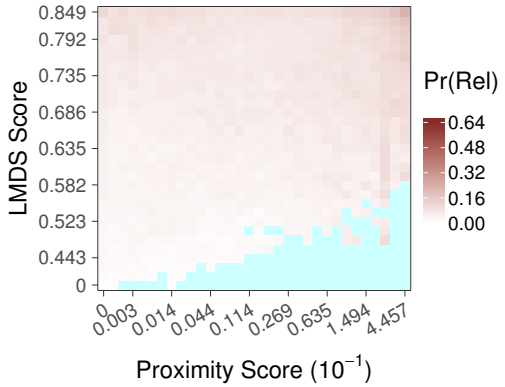
(1) For each of the queries Q in the query set, and each judged document D, calculate *FI-Score*(Q, D) and *P-Score*(Q, D).

(2) Divide each of *P-Score*(Q, D) and *FI-Score*(Q, D) into 32 equal ranges after normalizing the score ranges to $[0, 1]$ on a per-query basis, giving 1024 bins in total.

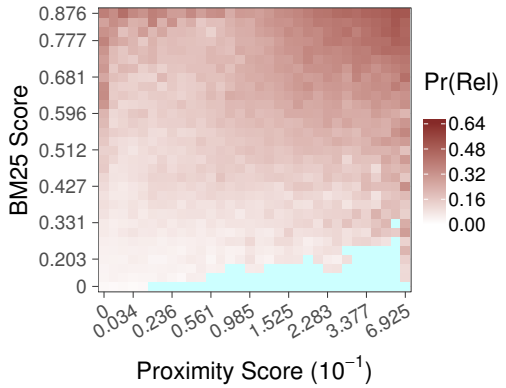
(3) For each bin $B_{i,j}$, estimate $\Pr(D = 1 \mid D \in B_{i,j})$ using the relevance judgment associated with each document in to that bin by any of the queries, each document may count repeatedly.



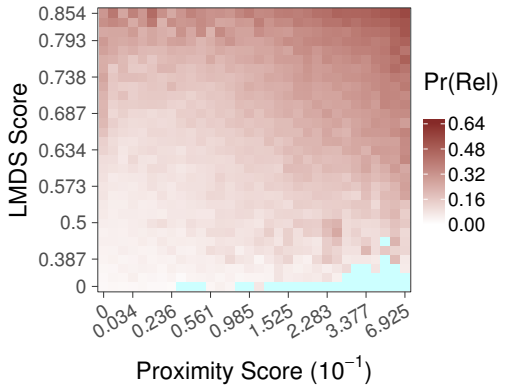
(a) BM25 and Proximity Score on Rob03.



(b) LMDS and Proximity Score on Rob03.



(c) BM25 and Proximity Score on TB04.

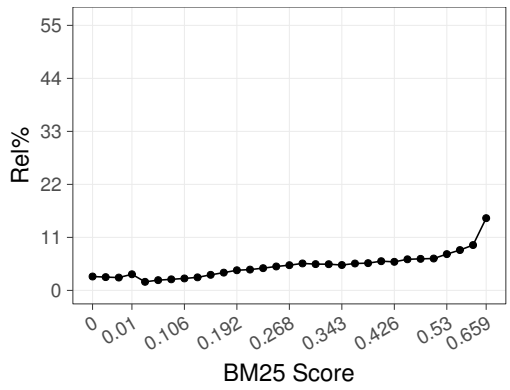


(d) LMDS and Proximity Score on TB04.

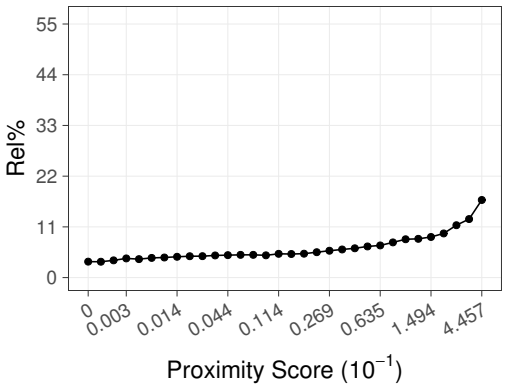
Figure 4.5: The relationship between bag-of-words scores and proximity scores. The top plots use the Rob03 queries and the judged documents from the TREC-8 collection; the second row plots the TB04 queries and the GOV2 document collection. The first column uses $\text{BM25}^{(2)}$ as the bag-of-words model and the second column uses LMDS (Equation 2.5) as the bag-of-words model. Proximity scores are computed using the scoring method defined in Equation 4.1 and 4.2. TREC title queries were used as well as the user queries. Measured document relevance in each of $1024 = 32^2$ bins is plotted, with blue cells having insufficient data to form estimates.

Results for the $\text{BM25}^{(2)}$ and LMDS model are plotted in Figure 4.5, with blue cells indicating bins with fewer than 100 documents, for which it would be misleading to estimate relevance probabilities; and with dark cells indicating regions of high probability. Relevant documents receive a zero score if none of the query terms appear in them, making the first bin bigger than the others.

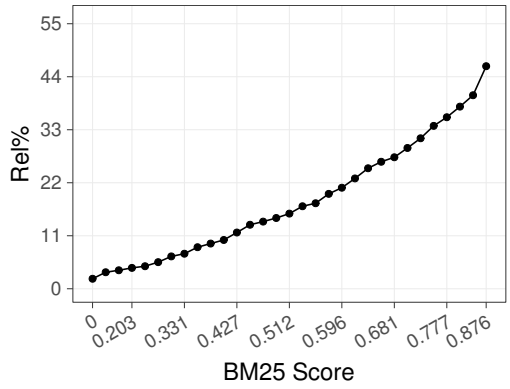
As expected, the higher the bag-of-words score, the higher the probability of relevance, and likewise for proximity scores. These two claims can be confirmed by projecting the bin values to the vertical and horizontal axes respectively, as shown in Figure 4.6. We consider BM25 as the bag-of-words model and plot the projection along both of the axes of Figure 4.5. It confirms our observation in the heatmap that either indicator can be used as evidence for document relevance. However, the heatmap in Figure 4.5 shows that using only one indicator may not be sufficient. If either of these factors were sufficient in their own right to predict relevance, with no influence



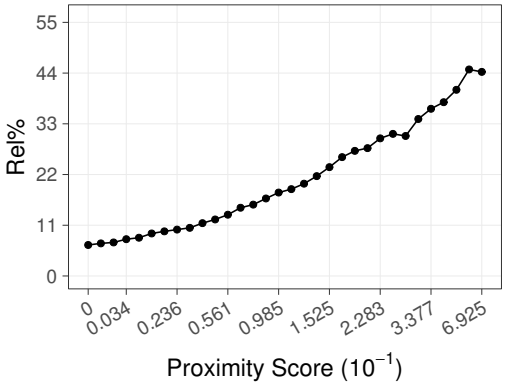
(a) BM25 Score on Rob03.



(b) Proximity Score on Rob03.



(c) BM25 on TB04.



(d) Proximity Score on TB04.

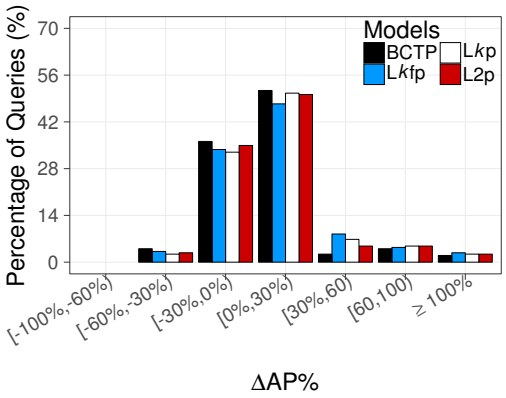
Figure 4.6: Projection of relevance percentages along each axis on both Rob03 (the first row) and TB04 (the second row) datasets. The first column projects the y-axis and second column projects the x-axis in Figure 4.5 (a) and (c). We omit results of LMDS since they are similar to BM25.

from the other, the patterns of shade in the graphs would be parallel to the corresponding axis. In fact, the darkest area in both of the heatmaps is located in the upper-right corner, suggesting that a combination of *FI-Score* and proximity score is a stronger indication of relevance than either of them is alone.

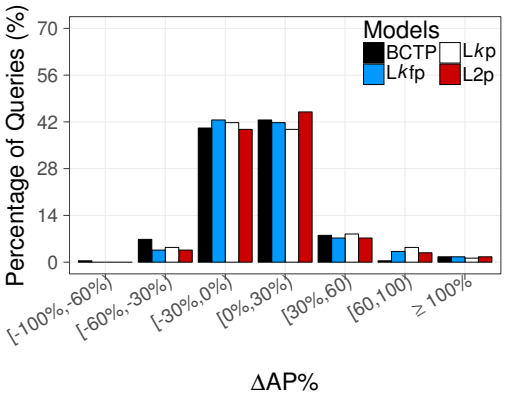
Note that the lighter color on Rob03 datasets compared to TB04 indicates a smaller background probability. As already noted, the percentage of relevant documents on Rob03 in the judgment set is smaller than the percentage on TB04 on average, thus the lighter overall color shown in Figure 4.5.

4.3.4 Retrieval Effectiveness using Local Proximity Statistics

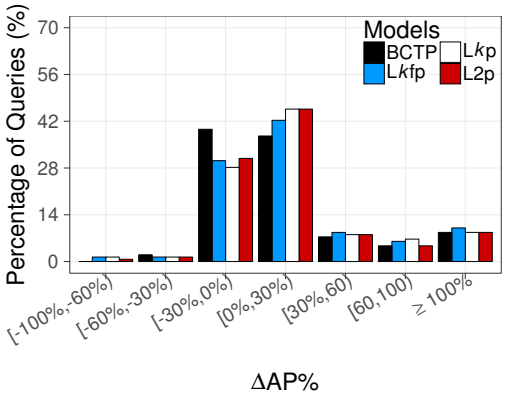
We analyze the proposed model on both NewsWire and large web datasets. As expected, there are cases where our methods improve and cases where all proximity-based retrieval models failed, we note these corner cases. The section ends with an exploration of trade-offs between efficiency and effectiveness.



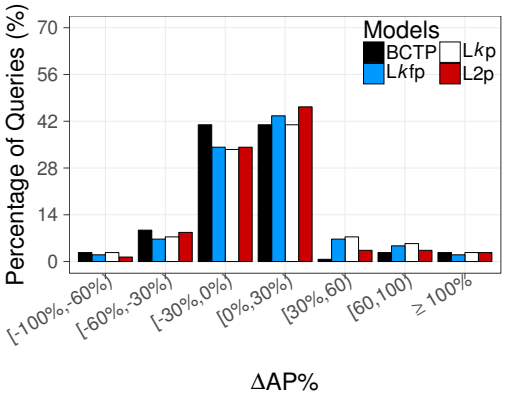
(a) ΔAP compared to BM25 on Rob04.



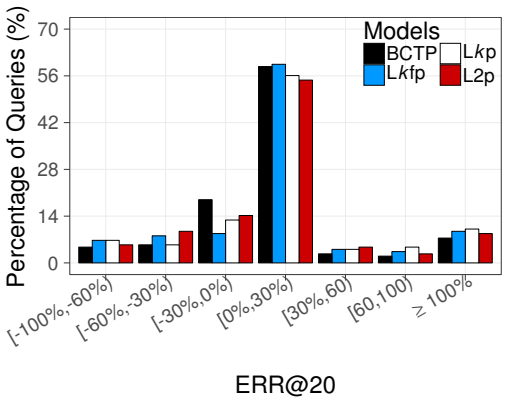
(b) ΔAP compared to SDM on Rob04.



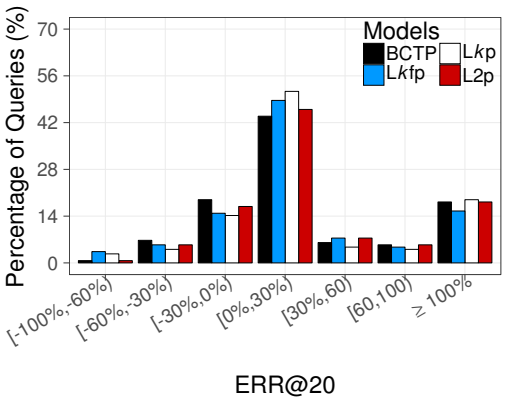
(c) ΔAP compared to BM25 on TB04–06.



(d) ΔAP compared to SDM on TB04–06.



(e) ΔERR@20 compared to BM25 on CW09B-2010–2012.



(f) ΔERR@20 compared to SDM on CW09B-2010–2012.

Figure 4.7: The left column shows the number of queries that fall into each of a set of percentage-difference buckets for each of Lkp, L$k$pf, L2p and BCTP, relative to the BM25 baseline run; and the right column shows the same computation carried out relative to SDM. The two top plots are for Rob04 measured using AP; the two middle plots are for TB04–06 also measured using AP; and the two bottom plots are for CW09B-2010–2012 measured using ERR@20. On all three collections, we only consider queries with more than one term. The total number of queries used in the comparisons are 238, 147, and 112, respectively across the three rows.

	Robust04		TB04–06	
	AP	RBP@0.95	AP	RBP@0.95
BM25$^{(1)}$	0.254	0.318	0.293	0.489
BM25$^{(2)}$	0.256	0.320	0.294	0.491
LMDS	0.248	0.308	0.290	0.474
SDM	0.262	0.322	0.319	0.506
FDM	0.264	0.322	**0.325**	0.512
BCTP	0.262	0.324	0.312	0.514
L2p	0.267	0.330†	0.320	**0.517**
Lkp	0.269	0.331†	0.321	0.515
L$k$pf	**0.270**†	**0.332**†	0.324	0.516
BestTrecRun	0.359	0.397	0.407	0.595

Table 4.4: Effectiveness of models on TREC-8 and GOV2, using two metrics. Statistical tests were performed, with † indicating significance relative to FDM at $p = 0.05$. Note that while the Robust04 test collection was also used as training to set the parameter λ used in Lkp, L$k$pf, and L2p (Table 4.3), the generated scores are relatively insensitive to the value used.

Effectiveness on Robust04 and TB04–06. Table 4.4 gives effectiveness scores for the models listed in Table 4.3, using the Robust04 and TB04–TB06 test collections respectively, with no stopping, and scoring based on runs of 1,000 returned documents. No RBP residuals (the score range allocated to unjudged documents) are shown in this table; but across the experiments listed, they were consistently around 0.03, because of the deep judgments available for these collections. The new L2p and Lkp methods are both better than the baseline BM25$^{(2)}$ model; the difference is significant at $p < 0.05$ when a two tailed t-test is performed, for both the AP and RBP@0.95 metrics. The higher-order proximity models also have a slight advantage compared to the bigram models, a relationship that was echoed in SDM and FDM. When including the additional distance constraint on Lkp to compute L$k$pf, observed effectiveness again increases very slightly. No training was undertaken in this regard, and further exploration may lead to a heuristic that obtains additional improvement. Huston and Croft [45] report an AP score of 0.329 on GOV2 for their WSDM-Int method, suggesting that applying additional resources to reweight subqueries may further improve performance.

Proximity improves performance on some queries and degrades it on others. The first column of graphs in Figure 4.7 plots the percentage change of score for Lkp compared to BM25$^{(2)}$, with queries grouped by the extent of the change in AP score measured on that query. All four of the methods generate more gains than losses, and hence give rise to overall improved performance. The three graphs in the second column of Figure 4.7 capture a similar comparison, but this time relative to the SDM baseline scores. The situation with regard to improvement is less clear cut for BCTP in the first two rows, reflecting the overall rates listed in Table 4.4, but in the third row, for the CW09B-2010–2012, all four of the plotted methods show an advantage.

Despite the overall benefit of proximity-based methods, up to a third of queries show worse effectiveness using Lkp than they do using the BM25$^{(2)}$ bag-of-words approach. Table 4.5 lists the

	Query	Lkp		Lkp	FDM
		AP	RBP@0.95	ΔAP%	ΔAP%
TB04-749	puerto rico state	0.037	0.070 + 0.01	−78%	−54%
TB04-720	federal welfare reform	0.080	0.223 + 0.01	−70%	−46%
R04-628	us invasion of panama	0.109	0.124 + 0.00	−56%	−39%
R04-381	alternative medicine	0.027	0.083 + 0.00	−54%	−69%
TB06-832	labor union activity	0.084	0.431 + 0.19	−47%	−17%
R04-389	illegal technology transfer	0.014	0.156 + 0.02	−45%	−50%
R04-437	deregulation gas electric	0.002	0.002 + 0.06	−41%	−8%
TB05-799	animals in alzheimer s research	0.184	0.324 + 0.05	−41%	−68%
R04-448	ship losses	0.012	0.055 + 0.03	−33%	−51%
R04-413	steel production	0.053	0.124 + 0.01	−33%	−17%
	Query	Lkp		Lkp	FDM
		AP	RBP@0.95	ΔAP%	ΔAP%
TB06-849	scalable vector graphics	0.325	0.489 + 0.15	+4300%	+5300%
TB05-825	national guard involvement in iraq	0.148	0.327 + 0.12	+500%	+400%
TB05-806	doctors without borders	0.486	0.594 + 0.03	+500%	+5%
R04-376	world court	0.108	0.442 + 0.04	+500%	+600%
TB06-829	spanish civil war support	0.289	0.305 + 0.03	+400%	+300%
TB06-828	secret shoppers	0.117	0.467 + 0.45	+300%	+1,000%
TB05-794	pet therapy	0.224	0.467 + 0.00	+200%	+500%
TB05-750	john edwards womens issues	0.337	0.260 + 0.11	+200%	+200%
R04-394	home schooling	0.139	0.126 + 0.01	+200%	+200%
TB06-844	segmental duplications	0.173	0.230 + 0.01	+200%	+300%

Table 4.5: The ten queries for which Lkp suffers the greatest degradation (the upper block) and improves the most (the bottom block) compared to BM25$^{(2)}$ on Robust04 and TB04–TB06 query sets, measured using percentage difference in AP score. The final column shows the degradation of FDM relative to BM25$^{(2)}$.

ten most degraded and improved queries when BM25$^{(2)}$ is compared to Lkp, with effectiveness difference measured by percentage loss in AP. While there is no obvious common element to these queries, if one could be identified based on index-resident information (such as IDF values), then a hybrid system that "dialed down" the value of λ (Equation 4.3) for certain queries could, potentially, be better than both BM25$^{(2)}$ and Lkp. This is an area for future investigation.

Effectiveness on CW09B Test Collections. Table 4.6 lists retrieval effectiveness outcomes using the three ClueWeb query sets 2010–2012; and Figure 4.7(e)(f) show the query improvement breakdown relative to BM25$^{(2)}$ and to SDM, respectively, aggregated across the same three query sets. The BM25 baseline performs very well on the first of the three query sets when measured using ERR, and BCTP works well on the other two sets when measured using RBP. Part of the explanation for the difference compared to the Rob04 and TB04–06 outcomes is that these queries contain more common words; the ClueWeb queries are also shorter overall, as is illustrated in Figure 4.4. Note also the relatively high RBP residuals observed for CW09B-2012 in particular; that

	CW09B-2010		CW09B-2011		CW09B-2012	
	ERR@20	RBP@0.8	ERR@20	RBP@0.8	ERR@20	RBP@0.8
BM25$^{(1)}$	0.112	0.230 + 0.10	0.130	0.241 + 0.13	0.125	0.200 + 0.18
BM25$^{(2)}$	**0.121**	0.236 + 0.02	0.166	0.312 + 0.02†	0.144	**0.212** + 0.12
LMDS	0.096	0.183 + 0.03	0.109	0.223 + 0.05	0.128	0.188 + 0.14
SDM	0.093	0.207 + 0.02	0.145	0.258 + 0.03	0.113	0.188 + 0.12
FDM	0.096	0.212 + 0.03	0.147	0.257 + 0.03	0.130	0.188 + 0.12
BCTP	0.105	0.222 + 0.02	0.162	**0.320** + 0.02†	0.155	0.210 + 0.10
L2p	0.117†	**0.247** + 0.03†	0.150	0.294 + 0.02	0.157	0.199 + 0.12
Lkp	0.114†	0.244 + 0.04	**0.173**	0.315 + 0.04†	0.164	0.201 + 0.13
L$k$pf	0.114†	0.237 + 0.03	0.158	0.308 + 0.05†	**0.166**	0.200 + 0.13
BestTrecRun	0.226	0.415 + 0.00	0.223	0.361 + 0.00	0.290	0.362 + 0.00

Table 4.6: Evaluation results of different models on ClueWeb09B, using queries from year 2010–2012, scored using ERR to depth 20 and RBP@0.8 using all available information. Statistical tests were performed, with † indicating significance relative to FDM at $p = 0.05$.

these are comparable in magnitude with the actual RBP scores is a warning that interpretation of these results needs to be undertaken with caution. Table 4.7 lists the most badly affected queries and the most improved queries when BM25$^{(2)}$ is compared to Lkp, now with "affected" determined by the percentage decrease in ERR. As with Table 4.5, there is no simple explanation as to why these queries suffer from the inclusion of proximity factors in the scoring regime. Note also – again, in common with Table 4.5 – that the FDM approach handles these queries just as poorly, and that the increased RBP residuals indicate that some of the loss of effectiveness may be a consequence of increased numbers of unjudged documents being retrieved compared to the BM25$^{(2)}$ runs. Were these documents to be judged, some of them might be relevant, lifting the Lkp scores. However, we also note that on some queries of CW09B test collections, while FDM performs worse than the BM25$^{(2)}$ baseline, the new proposed method shows an improvement in effectiveness.

4.3.5 Retrieval Cost Using Local Proximity Statistics

Table 4.8 lists per-document retrieval times. To compute these numbers, the time taken to score the top 1,000 answers for a query was measured, and then divided by the number of documents scored in order to obtain a per-document cost. In all of these methods the number of documents scored is taken to be the union of the query term's postings lists. Repeating that process across all of the queries in a collection allowed median, mean, and maximum values to be computed. Intervals for the L2p, Lkp, and L2pf were extracted using the PLANESWEEP approach introduced in the Chapter 3. The BM25 implementation is the fastest – it works solely with document-level statistics, and doesn't access term positional information. The L2p method is close behind, with times that are better than or the same as LMDS, and for the most part better than SDM. The Lkp approach is third overall in terms of speed, and it also executes more quickly than does SDM. As expected, FDM is slow – the cost of gathering the term statistics in a first pass is a substantial

Query		Lkp		Lkp	FDM
		ERR@20	RBP@0.8	ΔERR@20%	ΔERR@20%
C11-150	tn highway patrol	0.000	0.000 + 0.00	−100%	−100%
C12-161	furniture for small spaces	0.000	0.000 + 0.26	−100%	−100%
C12-169	battles in the civil war	0.000	0.001 + 0.26	−100%	−86%
C11-120	tv on computer	0.000	0.000 + 0.62	−100%	−74%
C12-184	civil right movement	0.000	0.000 + 0.68	−100%	−25%
C11-115	pacific northwest laboratory	0.010	0.017 + 0.08	−83%	−80%
C11-106	universal animal cuts reviews	0.010	0.015 + 0.02	−80%	−68%
C11-145	vines for shade	0.052	0.078 + 0.02	−78%	−83%
C10-59	how to build a fence	0.023	0.050 + 0.01	−73%	−80%
C11-114	adobe indian houses	0.030	0.086 + 0.03	−62%	−62%
Query		Lkp		Lkp	FDM
		ERR@20	RBP@0.8	ΔERR@20%	ΔERR@20%
C11-129	iowa food stamp program	0.133	0.211 + 0.00	+764%	+36%
C10-91	er tv show	0.191	0.595 + 0.00	+717%	+203%
C12-192	condos in florida	0.016	0.097 + 0.03	+290%	+72%
C12-157	the beatles rock band	0.015	0.058 + 0.18	+266%	+707%
C12-198	indiana state fairgrounds	0.949	0.346 + 0.00	+192%	−24%
C12-155	last supper painting	0.948	0.345 + 0.00	+160%	−65%
C10-71	living in india	0.117	0.350 + 0.22	+149%	+80%
C11-122	culpeper national cemetery	0.125	0.200 + 0.00	+118%	−100%
C11-140	east ridge high school	0.075	0.082 + 0.08	+100%	−100%
C12-190	brooks brothers clearance	0.078	0.175 + 0.00	+88%	−65%

Table 4.7: The ten queries for which Lkp suffers the greatest degradation (the upper block) and improves the most (the bottom block) compared to BM25$^{(2)}$ on the CW09B-2010–CW09B-2012 query sets, measured using percentage difference in ERR score. The final column shows the degradation of FDM relative to BM25$^{(2)}$.

burden. The high cost of computing large numbers of dependencies on long queries shows in the very high maximum times associated with FDM, and to a lesser extent, Lkp. Note that the measurement methodology favors SDM and FDM, since every document in which any query term appears is assumed to be scored, amortizing the cost of the first processing pass over a large number of evaluations. A more likely usage scenario would be to compute proximity statistics during a second phase, re-scoring a small subset of the documents that have been extracted using an efficient early stage candidate identification process. In this scenario the per-document costs shown in Table 4.8 remain accurate for the new methods, but are under-estimates for the SDM and FDM approaches. Dynamic pruning processes such as WAND, which reduce the number of documents scored, will have a similar effect.

Two further graphs conclude our presentation. Figure 4.8 plots per-document querying costs as a function of query length, and shows that SDM and FDM costs grow dramatically on long queries, and that L2p is only a little slower than BM25, at all query lengths. Finally, Figure 4.9 plots methods/collections in a trade-off graph in which the relationship between retrieval effec-

		BM25	LMDS	BCTP	SDM	FDM	L2p	Lkp
Robust04	Med	0.37	0.78	0.43	1.81	2.67	0.39	0.54
	Avg	0.36	1.25	0.46	1.81	2.77	0.39	0.60
	Max	0.53	1.23	1.42	9.63	15.48	0.63	2.25
TB04–06	Med	0.44	0.85	0.60	2.37	3.76	0.47	0.80
	Avg	0.44	0.87	0.80	2.68	6.84	0.48	1.05
	Max	0.67	1.36	6.90	9.12	74.84	0.95	9.31
CW09B-2010	Med	0.40	0.67	0.43	1.17	1.18	0.43	0.45
	Avg	0.26	0.48	0.61	1.58	3.98	0.30	0.92
	Max	0.67	1.35	8.89	16.65	73.28	1.32	12.95
CW09B-2011	Med	0.46	0.86	0.66	2.24	3.43	0.50	0.88
	Avg	0.48	0.93	0.95	2.76	6.50	0.54	1.34
	Max	0.65	1.25	8.27	14.69	32.59	1.24	11.17
CW09B-2012	Med	0.42	0.70	0.48	1.39	1.42	0.45	0.68
	Avg	0.34	0.76	0.71	1.99	4.21	0.39	0.98
	Max	0.60	6.97	7.24	14.16	59.60	1.17	10.20

Table 4.8: Retrieval cost in microseconds per document scored, using non-stopped indexes. In the CW09B-2010 results, Query 70 is dropped because of its extreme computation cost ("to be or not to be").

tiveness and retrieval cost can be seen clearly. Each collection is colored differently, and each shape corresponds to a retrieval model. The benefits of our Lkp (the solid circles) and L2p (the solid triangles) methods then become apparent – they provide high levels of retrieval effectiveness at speeds comparable to those attained by the bag-of-words BM25 mechanism.

4.4 Conclusions

We have investigated the use of higher-order proximity features in the retrieval process. One of the main past inhibitors has been cost, because of the need for global statistics. We have demonstrated the usefulness of integrating such higher-order proximity features and bag-of-words models for achieving better effectiveness. Our experimental results show that considering both is a stronger indicator of relevance than either of them alone.

As we have described, the careful combination requires us to make several design choices, when using MRFs to model the dependencies. The first design choice relates to what type of feature functions should be used, and the second is the choice of potential functions. Since the distance-based feature function typically requires global statistics at the term-level, we exploit this as a feature function, together with a BM25-like retrieval mechanism as the potential. The result compares favorably with FDM in terms of retrieval effectiveness, but executes substantially faster. In addition, it does not require global statistics, and hence can be applied in any pass of the retrieval process, without becoming proportionately more expensive.

The proposed method has several variants. When the same scoring computation is restricted

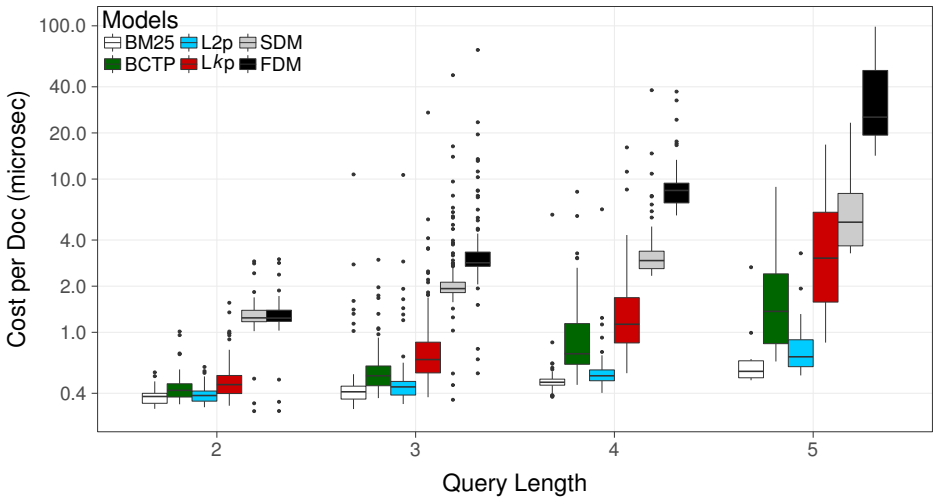


Figure 4.8: Per-document retrieval costs in microseconds, grouped by query length, with a logarithmic scale on the vertical axis. The measured query processing times are aggregated across all of the queries of lengths 2 to 5 in TREC title queries: Rob04, TB04–06, and CW09B-2010–2012. Unstopped indexes are used, and no query stopping.

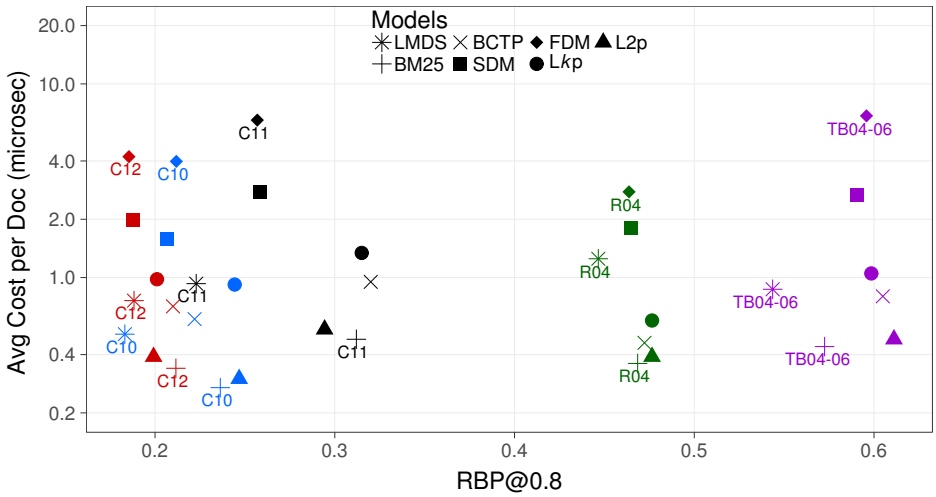


Figure 4.9: Trade-off graph for all retrieval models, across all collections, where "R" is "Robust", "T" stands for "TB" and "C" means "CW09B". Time is measured as the mean cost per document scored in microseconds, and effectiveness using RBP@0.8.

to query bigrams, speed is close to that of the simpler bag-of-words BM25 approach. Moreover, a term-pair index [13] may be used to further improve the query evaluation time. It is worth noting that additional efficiency can be achieved by indexing some or all higher-order term-dependencies when subquery lengths go beyond two. This is an interesting problem in its own right, and several prior studies have proposed time-space trade-offs at indexing time. Our current approach requires no additional space beyond the inclusion of positional information at indexing time; how to strategically employ further index space is an area for future work.

CHAPTER 5

The Effect of Pooling and Evaluation Depth

Collection-based evaluation of IR systems is an important aspect of system comparison. Often, the evaluation process requires to have settings that include a metric and a predefined evaluation depth. For an offline evaluation that makes use of existing test collections, pooling depths are also critical information to be aware of. In practice, the ranked list that is returned by retrieval systems is often limited in length. A fixed length of $1{,}000$ was employed in most TREC tracks, but some systems also return a length of $10{,}000$ in recent web tasks. Computing evaluation metrics over the original ranked list is commonly used. Recent emphasis on the importance of top-ranked documents leads to the use of truncated evaluation, which is usually undertaken over a relatively short prefix of a run – to an evaluation depth of perhaps $k = 10$, or to $k = 20$. An important question must then be considered: *to what extent are the various metric scores consistent, or at least correlated, as the evaluation depth k is varied*? Indeed, an even simpler question is this: What is it that is being measured when a metric that motivated as measuring the quality of a whole ranking is only applied to a prefix of it?

We have seen in Chapter 2 that a metric, especially a recall-based metric, may have different interpretations, and weighted-precision metrics are defined consistently regardless of the evaluation depth. A natural question that arise is how can the conclusions be consistent with each other when various evaluation depths are considered.

In the following sections, one of the main themes is to investigate the impact of evaluation depth k on truncated evaluation metrics. We show that the evaluation depth is a critical parameter in experimental IR evaluations, and plays a role that is inextricably inter-connected with the choice of metric itself. In particular, with a weighted-precision metric there is a clear sense that as k increases, the metric's value converges to a final value and that a to-depth-k evaluation is thus an approximation to the final true value of the metric. However, it is unclear from the metric definition whether the same conclusion can be made for recall-based metrics, and we empirically explore this for recall-based metrics in this chapter.

As mentioned, we need to be careful about choosing an evaluation setting according to specific test collections, or more specifically, the size of the judgment set. One of the most important indicators of judgment size is the pooling depth, which is directly related to the evaluation depth,

this chapter also examines how the combination of an evaluation depth k and a pooling depth d will affect the conclusion of system comparisons. Moreover, due to the different behaviors of recall-based and weighted-precision metrics, it is also interesting to know how well they are correlated with each other on test collections with different properties.

In order to compare different metrics, meta-evaluation approaches are required, and this is one of the main topics studied by many researchers in the field of evaluation. This chapter also considers the meta-evaluation approaches, since one of the main topics of this chapter is to gain a deeper understanding of the behavior of recall-based and weighted-precision metrics; to interpret how they differ from one another; and to figure out where they tend to agree with each other. Chapter 2 gave descriptions of various meta-evaluation methods for evaluation metrics, including discrimination ratio and rank correlation. In addition to these metrics and in order to be able to measure the agreement among metrics at various evaluation depths, we propose *coverage* and *inversion* ratio in this chapter. Also, to better grasp the correlation of metrics at various evaluation depths, instead of fixing one reference depth, this chapter also defines the *volatility* matrix, which shows an overview of the correlations between two metrics when varying evaluation depth k.

The sections are organized as follows: Section 5.1 introduces the relationship between the pooling depth and the evaluation depth, giving an overview of how the two variables are connected. In Section 5.2, we propose three meta-evaluation metrics, taking not only the rank correlation but also the discrimination ratio into consideration. The *volatility matrix* proposed in this section considers multiple references for differentiating between metrics. As mentioned, the pooling depth plays an important role in choosing a proper evaluation configuration, so we also give some observations on the widely used TREC datasets. A straightforward observation on the impact of different configurations of evaluation settings are explored in Section 5.4, directly showing the effectiveness scores of systems. Section 5.5 and Section 5.6 then provide interpretations of the observations via the proposed meta-evaluation approaches. Finally, we conclude the chapter in Section 5.7.

5.1 Evaluation Depth and Pooling Depth

The evaluation depth used in truncated metrics may or may not be the same as the pooling depth d. We start by considering only the n contributing systems, as shown in Figure 2.6 (Page 27). In those systems, documents that ranked higher than or equal to d are fully assessed. Two cases with regard to the pooling depth d can thus be differentiated: (i) $k \leq d$ and (ii) $k > d$. The second case is often known as *extended evaluation* [123].

It is obvious that an evaluation to depth $k \leq d$ using weighted-precision metrics only exploits the judged documents, while an extended evaluation needs to deal with documents that have not been judged. The same process using recall-based metrics might be viewed as an approximation of system effectiveness over the top-k documents. That is, an extra judgment is not helpful to the evaluation process of calculating system effectiveness score to depth k using weighted-precision metrics, but may affect the results of recall-based metrics. This is because adding extra judgments may provide a different estimate on the recall component. If more relevant documents can be

found in a subsequent judgment process, it usually, but not always, decreases the scores generated for $\text{AP}_a@k$, $\text{AP}_b@k$, $\text{NDCG}_a@k$ and $\text{QM}@k$. Only $\text{AP}_c@k$ and $\text{NDCG}_b@k$ remain unchanged, because the total number of relevant documents is fixed based on the top-k. The second case is $k > d$, and still, only contributing systems are considered. In this scenario, documents up to rank k are guaranteed to have relevance labels, but documents at depths deeper than d are likely to have unknown relevance values, resulting in an estimated effectiveness score instead of an actual one. Uncertainties associated with weighted-precision metrics can always be bounded using residual values.

The case of evaluating non-contributing systems requires careful consideration. This scenario is more complex than the previous two, because the unjudged documents, may also occur at ranks $k \leq d$, as well as $k > d$. A critical step in establishing the aptness of the experimental methodology should then be to establish the extent of the mismatch between test environment and systems being tested. In particular, if a new non-contributing system is generating runs in which only a minority of the top-k documents are judged, then the mismatch between system and testbed must be regarded as being suspect, and any scores that are computed must be handled very carefully.

5.2 Proposed Metric Comparison Methods

Section 2.4.5 introduced widely adopted approaches for metric comparisons. They can be applied to compare metrics with different evaluation depths. However, it is crucial for us to capture changes of particular systems in order to measure the behavior of a similar metric evaluating to different depths k. Therefore, *coverage* and *inversion* ratios are proposed in this section with an emphasis on capturing the significance changes among the same set of systems. In order to measure how well a metric can predict system orderings at different evaluation depths, the *volatility matrix* is proposed in Section 5.2.2.

5.2.1 Coverage and Inversion Ratio

Coverage Ratio. One issue with the use of discrimination ratio is that the two metrics can have the same discrimination ratio over a set of systems, but regard quite different pairs of systems as being distinct, or even regard the same pair of systems as being distinct but in two opposing directions. For example, Ravana and Moffat [83] measure the number of ordering reversals that take place as the pooling depth is increased. If the metrics are different, then such outcomes are not problematic, and can be expected to occur when metrics place their emphasis at different points in the ranking. But if the metrics are intended to be surrogates for each other, or are presumed to be strongly related, then such outcomes indicate a lack of reliability.

To capture this idea, metric $\text{M}^*(\cdot)$ is assumed to be a baseline, or a reference metric; and it is assumed that a second metric $\text{M}_{new}(\cdot)$ is being evaluated. Consider a set of systems $\{S_1, S_2, \ldots, S_m\}$ that has been scored using both $\text{M}^*(\cdot)$ and $\text{M}_{new}(\cdot)$. Suppose that $\text{M}^*(\cdot)$ discriminates a set of systems is defined

$$C_{ref} = \{(i, j) \mid \text{M}^*(S_i) >_p \text{M}^*(S_j)\}, \tag{5.1}$$

where $>_p$ indicates the use of the chosen statistic at test significance level p. Consider the evaluation results conducted using the new metric, similarly, the same significance level p as the reference metric is chosen and let C_{new} be the discriminative pairs, which is:

$$C_{new} = \{(i,j) \mid \mathrm{M}_{new}(S_i) >_p \mathrm{M}_{new}(S_j)\}. \tag{5.2}$$

Based on the two sets, we define the coverage ratio of $\mathrm{M}^*(\cdot)$ relative to $\mathrm{M}_{new}(\cdot)$ to be:

$$covrat_{\mathrm{M}^*}(\mathrm{M}_{new}) = \frac{|C_{ref} \cap C_{new}|}{C_{ref}}. \tag{5.3}$$

That is, the coverage ratio is the extent to which the new metric is able to confirm the relationships that were noted by the reference metric. For example, suppose that there are 10 systems to be compared and hence 45 distinct system pairs; that 35 of those system pairs are identified as being different at $p = 0.05$ by M^*; and that there are 28 out of the 35 that are also separated at $p = 0.05$ by M_{new}. Then the discrimination ratio of M^* is $35/45 = 0.78$, and the coverage ratio of M_{new} relative to M^* is $28/35 = 0.80$. The coverage ratio should be viewed as a modified discrimination ratio that is conditioned on the set of pairs that are found to be separable by the reference metric. In particular, if a metric that requires shallow judgments is intended as a surrogate for an established reference metric that requires a larger volume of judgments, then ideally, the coverage ratio between the two metrics should be high.

Inversion Ratio. Two metrics might also disagree when comparing systems. We define *observed inversions* with respect to the set of systems that is found to be significant using M^*, but M_{new} disagree with. That is, if $S_i >_p S_j$ is concluded using M^* at significance level p, but $S_i < S_j$ is obtained using M_{new}, then an inversion pair has been identified. Notice that when considering the inversion, there is no significance level required when measuring using M_{new}. We adopt a similar notation as for coverage and denote *inversion pairs* as:

$$I_{new} = \{(i,j) \mid \mathrm{M}_{new}(S_j) > \mathrm{M}_{new}(S_i) \text{ and } \mathrm{M}^*(S_j) <_p \mathrm{M}^*(S_i)\}.$$

Then the *inversion ratio* is computed as:

$$invrat_{\mathrm{M}^*}(\mathrm{M}_{new}) = \frac{|I_{new} \cap C_{ref}|}{C_{new}} \tag{5.4}$$

Continuing the same example with 10 systems and $|C_{ref}| = 35$, if 6 of those 35 system pairs result in $\mathrm{M}_{new}(s_j) > \mathrm{M}_{new}(s_i)$, then the observed inversion ratio would be $6/35 = 0.17$.

Interpretations of Coverage and Inversion Ratios. No particular conclusion can or should be drawn from the coverage ratio or the observed inversion ratio in cases in which the two metrics being compared are known to have different properties. Note that the two ratios are different from the *swap ratio* proposed by Buckley and Voorhees [14], where no statistical testing was involved. The very nature of statistical testing also means that some level of inversions and some loss of coverage can be expected, even when the metrics are closely related. But when a limited-depth evaluation is being used to generate an at-k approximation to a full-depth metric, such as via

the AP$_a$@k computation shown in Equation 2.31 (with d regarded as being a constant), or via the RBP(p)@k computation described, low coverage ratios and/or high observed inversion ratios should be regarded as a warning that the evaluation process may not be providing the desired measurement of system performance.

5.2.2 Volatility Matrix

We are interested in investigating how different evaluation depths affect the process of computing an effectiveness score of a metric. In order to explore this effect, the volatility matrix is proposed as a means of visualizing the strength of the relationship at different evaluation depths. Suppose that an M@k evaluation of a set of systems S results in (again, using an appropriate score aggregation regime over a set of topics, such as taking the mean) an ordering of the systems from highest overall score to lowest overall score given by the permutation $\sigma_k(S)$. Then for different depth k_1 and k_2, a matrix of correlation coefficients can be constructed, where

$$T_M[k_1, k_2] = \mathit{corr}(\sigma_{k_1}(S), \sigma_{k_2}(S))\,, \tag{5.5}$$

and where $\mathit{corr}(\cdot)$ is one of the rank correlation coefficients described earlier in this section. In the experiments presented in the next section, Kendall's τ is used as $\mathit{corr}(\cdot)$; the other rank correlation coefficients such as τ_{ap} and RBO can also be used as other visualization forms of the matrix. Assuming that $\mathit{corr}(\cdot)$ is a symmetric function, then T_M is a symmetric matrix, with all diagonal elements equal to 1. Taking a Kendall's τ score of 0.9 as a threshold for equivalence of system orderings [115], it would be ideal to expect that $T_M[k_1, k_2] > 0.9$ for all $k_1, k_2 > \theta$ for some threshold depth θ that depend both on the metric M and the set of systems S.

5.3 Observations of TREC Pooling Methods

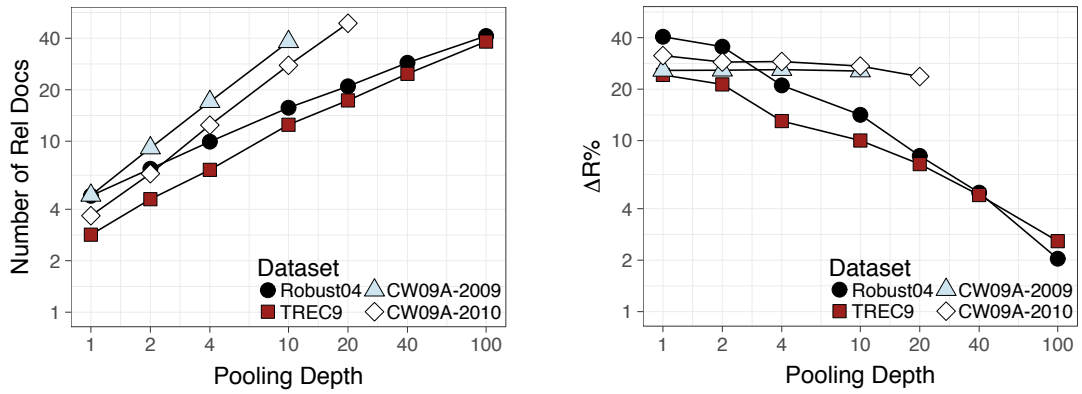
We explore how the different evaluation configurations affect the system comparison results in this section, describing widely used test collections. The experiments use contributing systems only, unless otherwise specified.

5.3.1 Collections, Topics and Systems

As mentioned in Section 2.5, early TREC test collections have relatively small sizes compared to the recent ClueWeb test collections. Conclusions may differ due to the difference in the test collections, so we consider both types in the experiments and use four test collections, as listed in Table 5.1. A detailed description of documents used for each test collection can be found in Section 2.5. As for the query sets, there are 250 topics in Robust04 test collections, and we choose to use the last 50 topics in our experiments because only this subset of topics are equally pooled to a depth $d = 100$. The TREC-9 web task uses a small web document collection and all topics are equally pooled to depth 100, though with less contributing systems compared to the Robust04 dataset. The judgment pool of CW09A-2009 was constructed using stratified sampling in which the first stratum is a $d = 12$ pool, and in CW09A-2010 test collection, the judgments are formed

Collection	Pooling depth	Contributing runs	Topics
Robust04	100	42	651 – 700
TREC-9 Web	100	25	451 – 500
CW09A-2009	12	32	1 – 50
CW09A-2010	20	21	51 – 100

Table 5.1: Descriptions of test collections used in the experiments, including pooling depth, number of contributing runs and considered topics. Note that on CW09A-2009, an additional sampling process is adopted in selecting documents to be judged and only the maximum "fixed-d" pooling depth across topics is reported and used.



(a) Number of relevant documents found.

(b) Percentage of new relevant documents.

Figure 5.1: Statistics of relevant documents and pooling process on different test collections. Figure (a) shows the cumulative number of relevant documents found when different pooling depths are employed; Figure (b) shows the additional percent of relevant documents found when increasing the pooling depth. Both figures are plotted in log-scale.

using a $d = 20$ pool. Although the last two datasets are much larger, their pooling depths are much shallower compared to the two earlier test collections. In order to observe the difference between the two types of test collections, shallow pooling depths of $d' < d$ are simulated by using top d' documents submitted by contributing systems. This simulation method also has been adopted by Zobel [134].

5.3.2 Relevant Documents

This section empirically examines the extent to which the number of relevant documents can be approximated on large test collections. Figure 5.1 shows the distribution of relevant documents found in the pooling process across the four collections. Figure 5.1(a) shows the number of documents with respect to different pooling depths and 5.1(b) shows the percentage of newly found relevant documents with an increasing pool depth, denoted as $\Delta R\%$. Both figures are computed by averaging across the contributing systems and topics listed in Table 5.1.

For both CW09A-2009 and CW09A-2010 datasets, more than a quarter of the ClueWeb documents that are inspected as the pool depth increases from $d = 4$ to $d = 10$ are found to

Dataset	Mode	RelInPool	RankUnj	NumRelPast
CW09A-2009	$d = 10$	2.4±3.0	12.9±2.6	13.6±12.7
CW09A-2009	$d = 12$	3.0±3.5	15.1±2.9	15.5±14.5
CW09A-2009	all	n/a	16.8±5.2	n/a
CW09A-2010	$d = 10$	3.3±3.2	12.3±2.1	17.2±13.2
CW09A-2010	$d = 20$	6.3±6.0	22.7±3.0	29.1±22.0
CW09A-2010	all	n/a	24.4±5.0	n/a

Table 5.2: Details of pooling outcomes, averaged across both systems and topics, with standard deviations. The column headed "RelInPool" is the average number of relevant documents per run identified within the pool construction depth indicated by the column headed "Mode"; "RankUnj" is the average rank of the first unjudged document; and "NumRelPast" is the average number of identified relevant documents that appear beyond the listed pool construction depth. The average number of relevant documents per topic at the given value of d is the sum of "RelInPool" and "NumRelPast".

be relevant. These observations suggest that there remain a lot of relevant documents unjudged, which is in marked contrast to the Robust04 and TREC-9 dataset. Rather, it seems likely that a further one-fifth of the unjudged documents that occur in the top 50 of the contributing runs could be relevant, doubling the number of known answers. Table 5.2 provides further details of the effect of different pool depths for the two CW09A test collections. There may be many reasons for explaining a higher percentage of relevant documents in the web test collections, for example, topic difficulty. We do not consider factors other than pooling depth in our experimental studies. The table lists "RelInPool", which is the average number of relevant documents per run per topic identified within the pool construction depth indicated by the column labeled "Mode". The column "RankUnj" in the table is the average rank of the first unjudged document; and "NumRelPast" is the average number of relevant documents that appear beyond the listed pool construction depth. The average number of relevant documents per topic at the given value of d is the sum of "RelInPool" and "NumRelPast". Statistics listed in Table 5.2 further confirm the observation made in Figure 5.1, which indicates the insufficient judgment pool on all ClueWeb datasets.

In addition to investigating how relevant document statistics change when pooling depth varies, experiments have been conducted to observe the vulnerability of the pool relative to the number of contributing systems. The results of leave-one-out experiments are listed in Table 5.3. We simulate a shallow pool on Robust04 test collection using $d = 10$, and keep the original pooling depth on the other two collections, which are $d = 12$ and $d = 20$ for CW09A-2009 and CW09A-2010, respectively. As noted, only contributing systems are considered in the experiments. Both percentage of score changes ($\Delta\%$) and absolute rank change ("ΔRank") are computed and listed in the table. The first block of the table shows the result of a leave-one-out experiment [134], and the second block presents results of a leave-group-out experiment [115]. It is clear that in both experiments, the two web collections are vulnerable to both score and system rank shifts, even when shallow metrics are used. Both experiments confirm that Robust04 test collection is more reusable than the web collections, that is, a new system can be more fairly evaluated using the set of judgments, but not on CW09A datasets. Similar observations have also

	Robust04		CW09A-2009		CW09A-2010	
	$\Delta\%$	ΔRank	$\Delta\%$	ΔRank	$\Delta\%$	ΔRank
AP_b	–1.31	0.67	–23.30	4.13	–17.50	2.43
$NDCG_a$	–1.06	0.60	–18.54	4.03	–13.98	2.10
$RBP(0.8)@k$	–0.90	0.38	–17.12	3.66	–12.75	1.61
$RBP(0.95)@k$	–1.25	0.76	–20.51	4.56	–15.33	2.19

(a) Score and rank changes resulting from leaving out one run.

	Robust04		CW09A-2009		CW09A-2010	
	$\Delta\%$	ΔRank	$\Delta\%$	ΔRank	$\Delta\%$	ΔRank
AP_b	–4.41	1.81	–48.78	9.57	–42.71	5.43
$NDCG_a$	–3.30	1.80	–41.79	8.69	–35.02	5.19
$RBP(0.8)@k$	–3.05	1.31	–40.51	7.75	–30.94	4.39
$RBP(0.95)@k$	–4.10	1.85	–44.07	9.07	–37.38	5.89

(b) Score and rank changes resulting from leaving out a related group of runs.

Table 5.3: Leave-one-run-out and leave-one-group-out experiments, with $k = d = 10$ for Robust04, $k = d = 12$ for CW09A-2009 and $k = d = 20$ for CW09A-2010, where $\Delta\%$ is the percentage score change, and ΔRank is the absolute rank change among the systems. Values listed are averaged over the left out run or runs.

been made on tasks using the GOV2 document collection [16].

5.3.3 Breaking Ties

One subtle but important point to consider when implementing evaluation metrics is how to break tied scores in a ranked list, and how different tie breaking strategies affect the evaluation results. This received little attention since ranked results are often assumed to be given in a deterministic order, which may not be the case in practice. Systems may submit a ranked list containing one or more tied scores, even at the top ranking positions. Only until recently has the effect of tied evaluation scores been carefully explored [126]. In order to have an overall understanding of the TREC pooling process, a wider setting of test collection is used in this section. Table 5.4 shows the median percentage of score ties within the top 20 documents per topic, across all submitted systems for a wide range of TREC tracks. The median per-run percentage of ties among the top 20 documents differs widely across tasks, and the potential pitfalls of ignoring it are quite real.

Tie breaking methods are often implicitly defined at an implementation level, and are built into the various evaluation tools. In general, there are three main approaches adopted. The first one is used in the official evaluation toolkit trec_eval , it breaks document score ties based on reverse alphabetical ordering of document labels. This approach, while resulting in a deterministic ranking, risks promoting unjudged documents higher in the overall ranking than may be warranted by their original position in the submitted run. In particular, if a document demoted by trec_eval as a result of a tied score had in fact been included in the pool and judged relevant, and a corresponding promoted document had been unjudged or judged non-relevant, then the system performance will be underestimated. To minimize this risk, through most (but not all) of the TREC experi-

Task	Med. % of ties	τ	Task	Med. % of ties	τ
TREC-2	12	0.742	Robust04	9	0.997
TREC-3	6	0.856	Robust05	64	0.990
TREC-4	14	0.838	TREC 2004	44	0.999
TREC-5	8	0.981	TREC 2005	30	0.990
TREC-6	8	0.997	TREC 2006	42	0.987
TREC-7	12	0.996	CW09A-2009	22	0.898
TREC-8	10	0.990	CW09A-2010	35	0.959
TREC-9	34	0.994	CW09A-2011	14	0.984
TREC-10	42	0.995	CW09A-2012	14	1.000

Table 5.4: The median percentage of tied scores, and the Kendall's τ correlation between system orderings generated by TREC-style tie-breaking and system orderings generated by document-occurrence based tie-breaking, evaluated using RBP(0.8). Only the top-20 documents are considered when computing the percentage of tied scores.

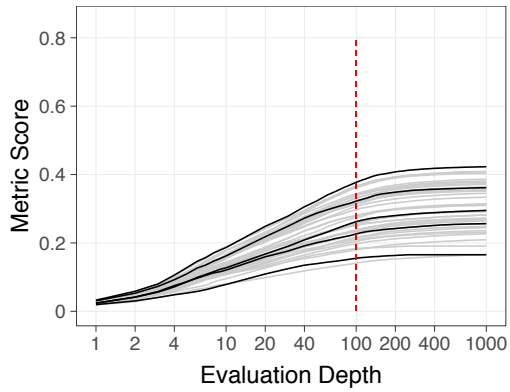
mentation, the pooling process was also implemented in a way that broke ties that straddled the pooling depth so as to ensure that the documents that would be included by trec_eval in an at-k evaluation would also be the ones judged as part of an at-k pool.

An obvious second alternative is to adhere to the ordering of the lines in the run file, and not reorder score-tied documents in any way, which is the approach used in the NTCIR evaluation tools that have been used since NTCIR-7. Doing so has the benefit of ensuring that any subsequent user-instituted pooling implemented using the equivalent of "head -k" corresponds exactly to the subsequent evaluation process.

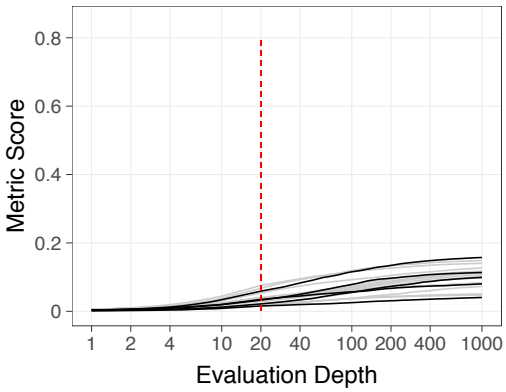
The third approach of handling ties depends only on the document scores, and it can only be applied with weighted-precision metrics. It computes score contributions across block of tied documents, summing the total weight available at that set of depths, and then sharing that total uniformly across the documents in that block. Each apportioned weight is then used to scale the corresponding per-document gain. A further variant is described by Moffat et al. [76], who suggest computing the average gain across the documents in the block of equal-score items, and then applying that gain at each rank position spanned by the block. Bevan Koopman's inst_eval[1] implements this suggestion.

Using the same evaluation metric but with different tie-breaking strategies might result in potentially unfair comparisons between systems. For simplicity, RBP(0.8) is adopted in Table 5.4. The reason to apply the weighted-precision metric in the experiments other than recall based metrics is to prevent potential side-effects caused by truncation. Two tie breaking methods are considered when calculating Kendall's τ in Table 5.4: (i) TREC-style, using reverse ordering of document IDs to break ties; and (ii) the adherence to original rankings submitted by each retrieval system. As shown in Table 5.4, in most of the tasks, Kendall's τ values indicate a high correlation between the two different approaches. However, there remains subtle inconsistencies in system ordering and an incorrect sorting may lead to some mistakes [126].

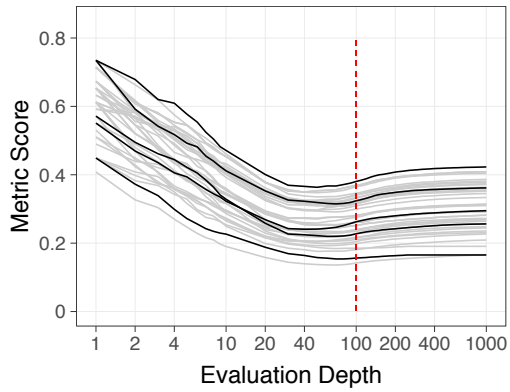
[1] https://github.com/bevankoopman/inst_eval



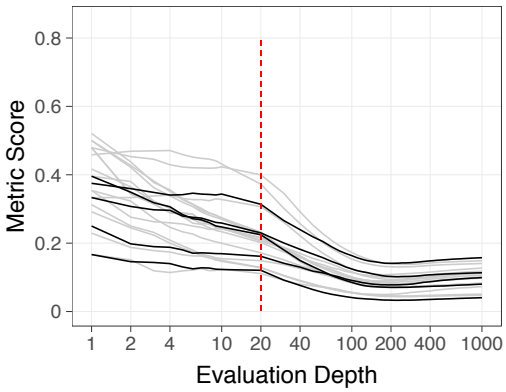
(a) $\text{AP}_a@k$ on Robust04, $d = 100$.



(b) $\text{AP}_a@k$ on CW09A-2010, $d = 20$.



(c) $\text{AP}_b@k$ on Robust04, $d = 100$.



(d) $\text{AP}_b@k$ on CW09A-2010, $d = 20$.

Figure 5.2: Raw system scores for AP@k variants, on both Robust04 and CW09A-2010. The first column shows results on Robust04 dataset and the second column shows results on the CW09A-2010 test collection. All the scores are computed over the set of topics and each line represent one submitted system. All figures are computed based on contributing systems.

For correctness and consistency, the second tie breaking method is used throughout in all following experiments, which considers only the document orderings submitted by a system, without referring to other external evidence. Also, unlike the third method that may only be applied with weighted-precision metrics, it can be used with any arbitrary metric. No matter which tie breaking strategy is employed, it is important to maintain a consistent strategy in both the pooling and evaluation process.

5.4 Observations of System Effectiveness Scores

The question of how different evaluation depths affect system comparisons is addressed in this section from a raw score perspective, considering both recall-based and weighted-precision metrics. Experiments in this section mainly make use of two datasets: the Robust04 and the CW09A-2010.

5.4.1 Effectiveness Scores of Recall-Based Metrics

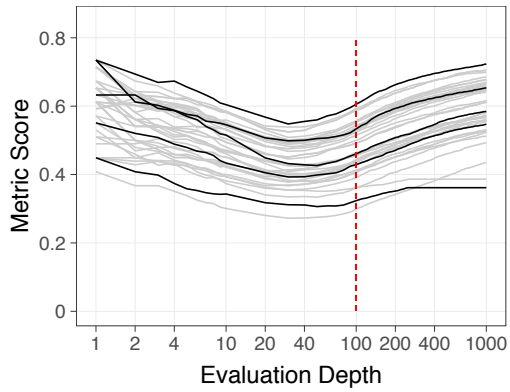
Truncated AP Variants. As described in Section 2.4, there are three variants of truncating AP to a predefined evaluation depth k, among which, Equation 2.31 and 2.32 are applied in two widely used evaluation tools. Figure 5.2 shows the difference between the two variants of truncated AP calculation, which plots system effectiveness scores as a function of the evaluation depth k (using a log scale). Each black line in the figure represents a contributing system, with five of the systems (the minimum, first quartile, median, third quartile, and maximum, all as of the right-hand edge of each graph) emphasized with darker lines so that the overall pattern can be discerned. We also plot vertical red lines for reference, showing the pooling depth d, so that when evaluation depth $k \leq d$, all documents are judged and when $k > d$, there are unjudged documents. The upper row of Figure 5.2 shows system scores calculated using $\text{AP}_a@k$, and the lower row shows evaluation results using $\text{AP}_b@k$. As a comparison, the evaluation results on both Robust04 and CW09A-2010 are depicted in the left and right column, respectively.

It is clear that when using $\text{AP}_a@k$, the scores will not decrease at all, but the same density of cross-overs arise; moreover the metric scores are compact in a tighter range than $\text{AP}_b@k$. The alternative normalization regime in the second column does not address the cross-over issue either. Moreover, $\text{AP}_b@k$ is not non-decreasing, and unpredictable with the increase of the evaluation depth. However, once the evaluation depth $k \geq d$, the two variants show the same behavior, due to the normalization factor reaching the same value, and because unjudged documents are always assumed to be not relevant in both calculations.

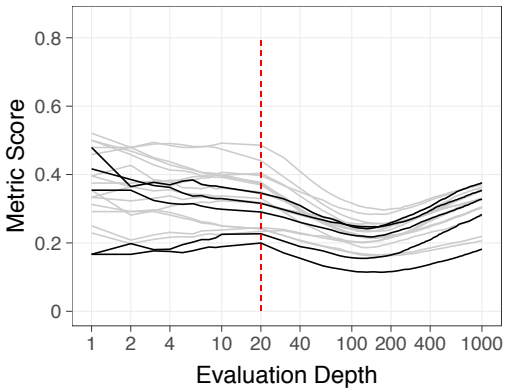
$\text{NDCG}_a@k$ and QM@k. The system scores evaluated using the other two recall-based metrics are shown in Figure 5.3. The results of $\text{NDCG}_a@k$ on the two collections are shown in the top row and results of QM@k ($\beta = 1$) are shown in the bottom row, with all the other arrangements being the same. Scores computed using $\text{NDCG}_a@k$ show the same unpredictable patterns as $\text{AP}_b@k$ in Figure 5.2, which contains cross-overs even when $k \leq d$. Because QM@k actually is an blend of $\text{AP}_b@k$ and $\text{NDCG}_a@k$, calibrated using parameter β, so it is unsurprising that it exhibits similar behavior to these two metrics.

5.4.2 Effectiveness Scores of Weighted-Precision Metrics

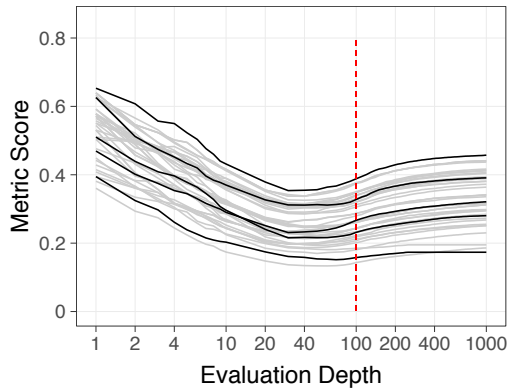
Effectiveness scores of systems calculated using weighted-precision metrics are presented in Figure 5.4, with same arrangement. The top row shows the $\text{RBP}(0.95)@k$ results and the bottom row shows system effectiveness scores computed using ERR@k. The major different pattern between the two metrics is straightforward. That is, the scores of $\text{RBP}(0.95)@k$ are relatively "stable" at a deeper expected pooling depth than ERR@k. This difference occurs because $\text{RBP}(0.95)@k$ is "deep" while ERR@k is shallow. The different choices of persistence values p can let RBP behave in a different way – a small p value makes RBP a shallow metric while a large p value adds more weight to relevant documents retrieved at the bottom of a ranked list. The different choices of discount functions of DCG@k (listed in Table 2.4) may have a similar effect. A discount function that gives a small weight to a relevant document at deep ranks would allow DCG@k to exhibit



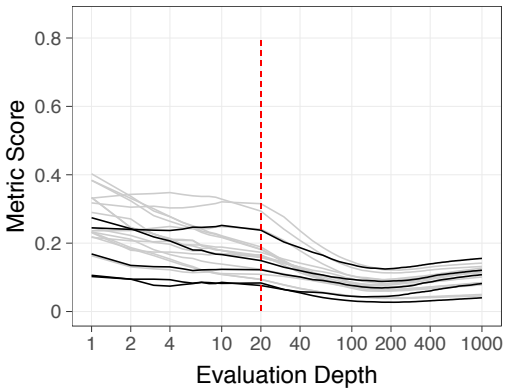
(a) NDCG$_a$@k on Robust04, $d = 100$.



(b) NDCG$_a$@k on CW09A-2010, $d = 20$.



(c) QM@k on Robust04, $d = 100$.



(d) QM@k on CW09A-2010, $d = 20$.

Figure 5.3: Scores for NDCG$_a$@k and QM@k on Robust04 and CW09A-2010, with the same experimenal settings as Figure 5.2.

a similar behavior to shallow metrics. A discount function which distributes more evenly would behave similar to deep metrics. When compared with recall-based metrics, more stable system orderings are observed as the evaluation depth is varied, including in the case when extended evaluation is employed. The same stable pattern of system orderings for recall-independent metrics is also observed on the other two test collections, in results that are not included here.

5.4.3 The Effect of Shallow Judgment Pools

Our previous discussion fixed the pooling depth as used in the original TREC experimental settings. Shallow pooling depths may also affect the evaluation outcome. In order to explore the effect, similar experiments are done on both Robust04 and CW09A-2010, but with simulated depths $d = 10$. Raw system effectiveness scores are shown in Figure 5.5. Results of both types of metrics are presented, with the same settings, but the only difference being the pooling depths. We set a smaller persistence value $p = 0.8$ in the RBP calculation to match the pooling depth.

We first compare between the two types of metrics. It is clear that the previous observations that were made are consistent: (i) the weighted-precision metric RBP(0.8)@k shows a relatively

k	$AP_b@k$			$QM@k$			$RBP(0.95)@k$		
	Disc.	Covr.	Invr.	Disc.	Covr.	Invr.	Disc.	Covr.	Invr.
1	**21.1**	31.1	5.2	20.8	30.3	7.4	**21.1**	33.7	5.3
4	40.9	58.7	2.1	39.5	56.0	1.6	**43.0**	67.2	0.4
10	51.9	75.3	**0.0**	51.7	74.7	0.7	**53.4**	84.0	**0.0**
40	63.2	93.2	**0.0**	**63.4**	92.8	**0.0**	61.3	98.8	**0.0**
100	66.9	**100.0**	**0.0**	**67.4**	**100.0**	**0.0**	61.0	**100.0**	0.0
400	67.9	99.3	**0.0**	**68.5**	99.8	**0.0**	61.0	**100.0**	**0.0**
1000	68.4	99.7	**0.0**	**70.0**	99.8	**0.0**	61.0	**100.0**	**0.0**

k	$RR@k$			$ERR@k$			$RBP(0.8)@k$		
	Disc.	Covr.	Invr.	Disc.	Covr.	Invr.	Disc.	Covr.	Invr.
1	23.5	100.0	**0.0**	13.4	**56.8**	**0.0**	**21.1**	40.3	3.6
4	23.5	100.0	**0.0**	20.0	**86.5**	**0.0**	41.5	81.1	**0.0**
10	23.5	100.0	**0.0**	22.8	**96.9**	**0.0**	49.0	95.4	**0.0**
40	23.5	**100.0**	**0.0**	22.3	**100.0**	**0.0**	51.0	**100.0**	**0.0**
100	23.5	**100.0**	**0.0**	22.3	**100.0**	**0.0**	51.0	**100.0**	**0.0**
400	23.5	**100.0**	**0.0**	22.3	**100.0**	**0.0**	51.0	**100.0**	**0.0**
1000	23.5	**100.0**	**0.0**	22.3	**100.0**	**0.0**	51.0	**100.0**	**0.0**

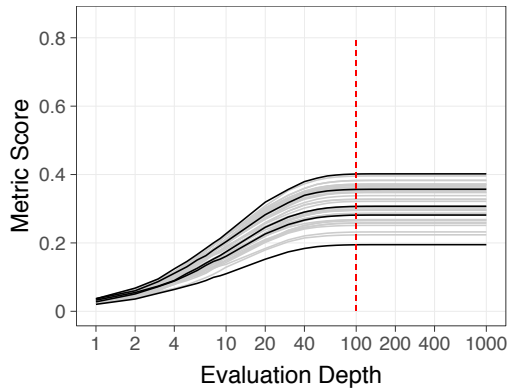
(a) Evaluation using the Robust04 collection, with $d = 100$ in all cases.

k	$AP_b@k$			$QM@k$			$RBP(0.95)@k$		
	Disc.	Covr.	Invr.	Disc.	Covr.	Invr.	Disc.	Covr.	Invr.
1	30.0	53.2	8.1	36.7	54.3	6.7	30.0	55.3	1.0
4	46.2	82.9	**0.0**	**48.6**	81.9	**0.0**	41.4	74.8	**0.0**
10	52.9	**100.0**	**0.0**	**50.0**	**100.0**	**0.0**	49.0	**100.0**	**0.0**
40	48.6	82.9	**0.0**	47.1	81.9	**0.0**	**49.0**	80.6	**0.0**
100	**49.5**	79.3	**0.0**	48.6	74.3	1.9	47.6	79.6	1.0
400	**51.0**	78.4	1.8	50.5	73.3	4.8	48.6	79.6	1.0
1000	**50.5**	77.5	1.8	50.0	72.4	6.7	48.6	79.6	1.0

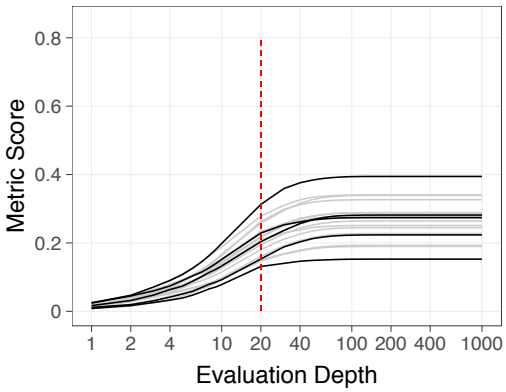
k	$RR@k$			$ERR@k$			$RBP(0.8)@k$		
	Disc.	Covr.	Invr.	Disc.	Covr.	Invr.	Disc.	Covr.	Invr.
1	41.4	100.0	0.0	**33.8**	**66.7**	**0.0**	30.0	63.4	**0.0**
4	41.4	100.0	**0.0**	**48.6**	**87.6**	**0.0**	40.5	82.8	**0.0**
10	41.4	100.0	**0.0**	**50.0**	**100.0**	**0.0**	44.3	**100.0**	**0.0**
40	41.4	100.0	**0.0**	48.1	96.2	**0.0**	45.7	**97.8**	**0.0**
100	41.4	100.0	**0.0**	48.1	96.2	**0.0**	45.7	**97.8**	**0.0**
400	41.4	100.0	**0.0**	48.6	96.2	**0.0**	45.7	**97.8**	**0.0**
1000	41.4	100.0	**0.0**	48.6	96.2	**0.0**	45.7	**97.8**	**0.0**

(b) Evaluation using the CW09A-2010 collection, with $d = 10$ in all cases.

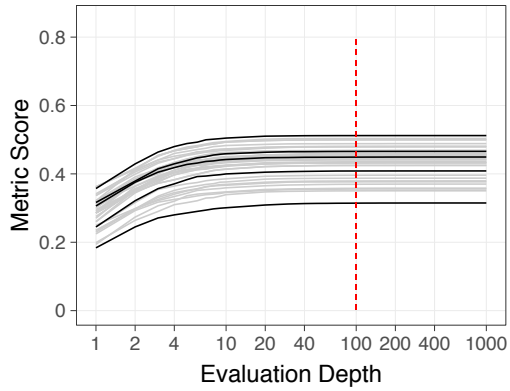
Table 5.5: Discrimination, coverage, and observed inversion ratios (all expressed as percentages at $p = 0.05$) for six metrics at different evaluation depths k, and two different document collections. The reference metric for the coverage and inversion ratios is the corresponding metric at depth $k = d = 100$ and $k = d = 10$, respectively (see Tables 5.1 and 5.2 for pooling details).



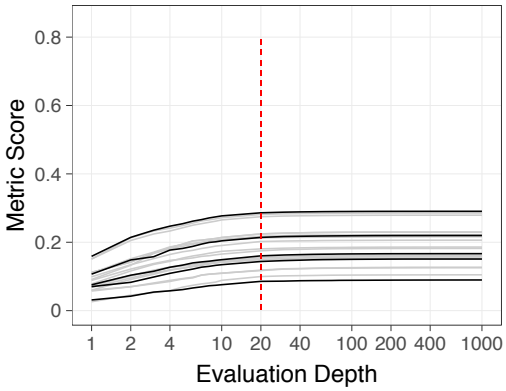
(a) RBP$(0.95)@k$ on Robust04, $d = 100$.



(b) RBP$(0.95)@k$ on CW09A-2010, $d = 20$.



(c) ERR$@k$ on Robust04, $d = 100$.



(d) ERR$@k$ on CW09A-2010, $d = 20$.

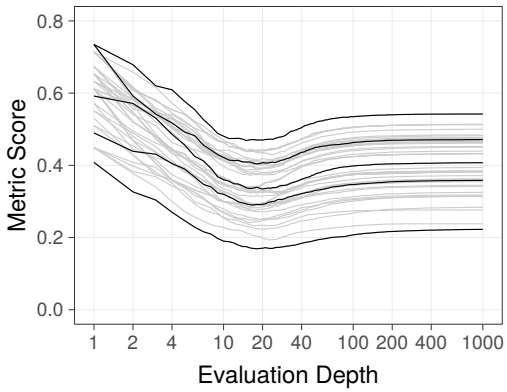
Figure 5.4: Scores for RBP$(0.95)@k$ and ERR$@k$ on Robust04 and CW09A-2010, with the same experimental setting in Figure 5.2.

stable system ranking compared to both AP$_b@k$ and NDCG$_a@k$; and (ii) the cross-overs in both recall-based metrics occur regardless of whether $k \leq d$. Considering Figure 5.2, 5.3 and Figure 5.5, it can be observed that a shallower pooling depth results in more volatile system rankings (in terms of cross-overs), especially for recall-based metrics such as NDCG$_a@k$ and AP$_b@k$.

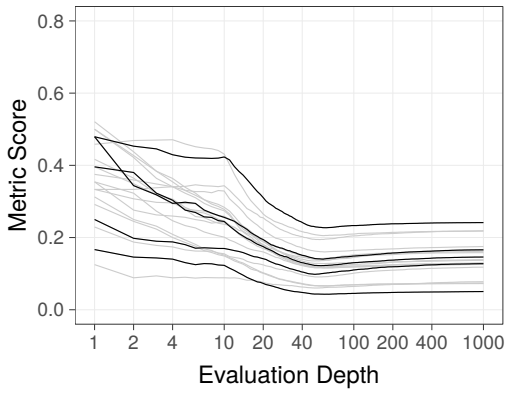
5.5 Meta-Evaluation with a Fixed Reference

It can be observed from the raw score figures that recall-based metrics are vulnerable to both evaluation and pooling depth effects. In order to quantify the impact of different choices of various metrics, only the evaluation depth is varied, with a fixed pooling depth across all experiments presented this section. The approaches discussed in Section 5.2 are considered, in order to provide evidence. In the process of computing the correlation methods, the evaluation depth $k = d$ is regarded as the reference point. Results of NDCG$_a@k$ are shown in Figure 5.6. Similar observations can be made on other recall-based metrics and are presented in Table 5.5.

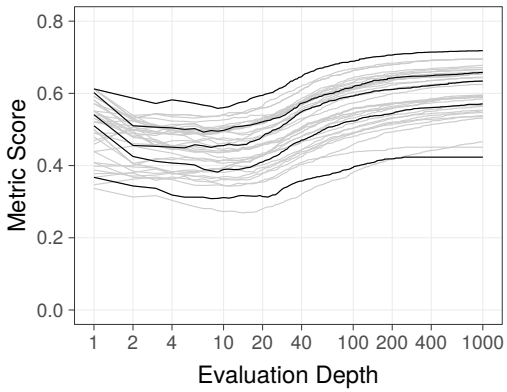
Four test collections (see Table 5.2) are used in Figure 5.6, all adjusted so that they are pooled to a uniform depth of $d = 10$; plus the two NewsWire datasets are also used pooled to their judged



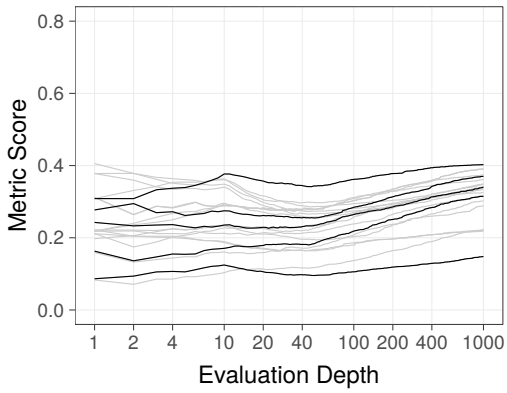
(a) $\text{AP}_b@k$ on Robust04.



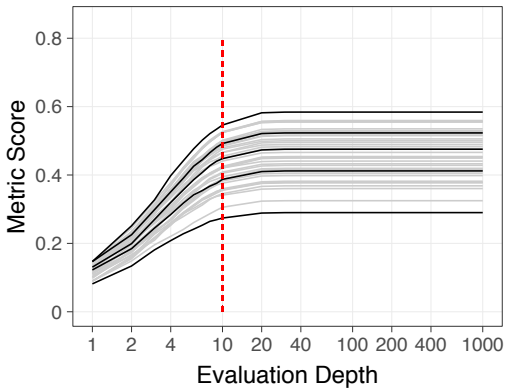
(b) $\text{AP}_b@k$ on CW09A-2010.



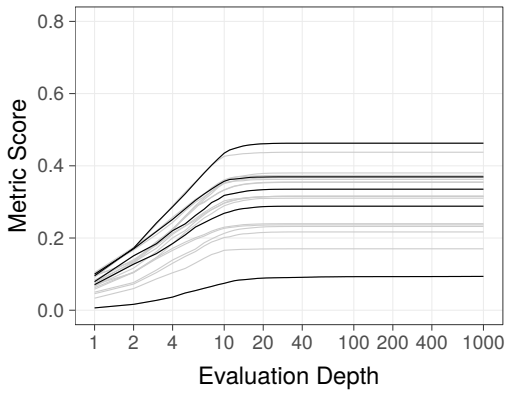
(c) $\text{NDCG}_a@k$ on Robust04.



(d) $\text{NDCG}_a@k$ on CW09A-2010.



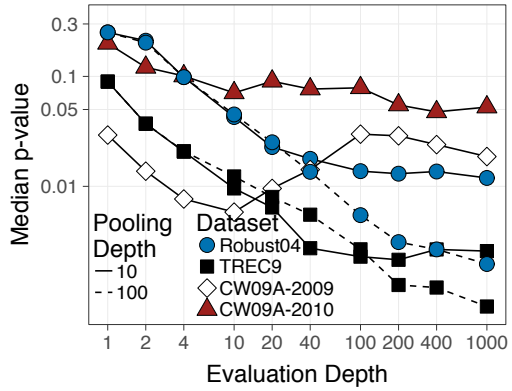
(e) $\text{RBP}(0.8)@k$ on Robust04.



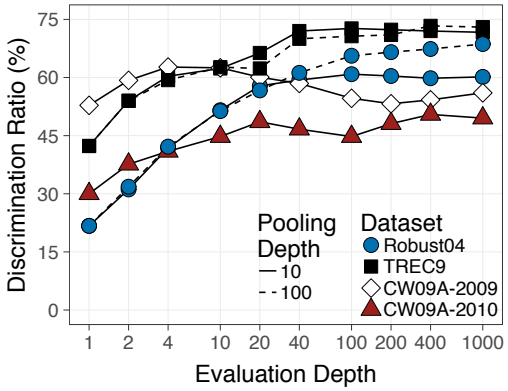
(f) $\text{RBP}(0.8)@k$ on CW09A-2010.

Figure 5.5: $\text{AP}_b@k$, $\text{NDCG}_a@k$ and $\text{RBP}(0.8)@k$ on Robust04 and CW09A-2010, $d = 10$.

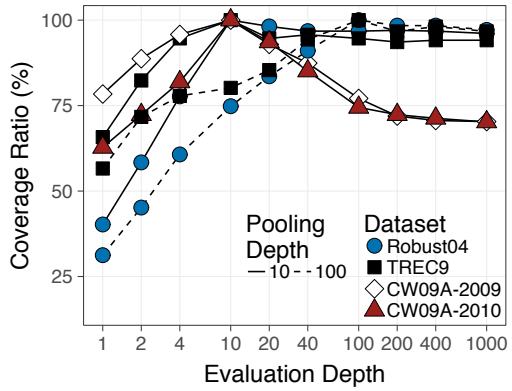
depth of $d = 100$. That is, there are six experimental contexts in total, four pooled to a depth of $d = 10$, and two pooled to a depth of $d = 100$. The six components in Figure 5.6 will be discussed in order.



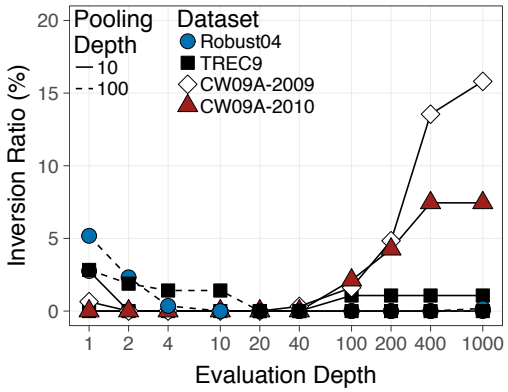
(a) Median p-values.



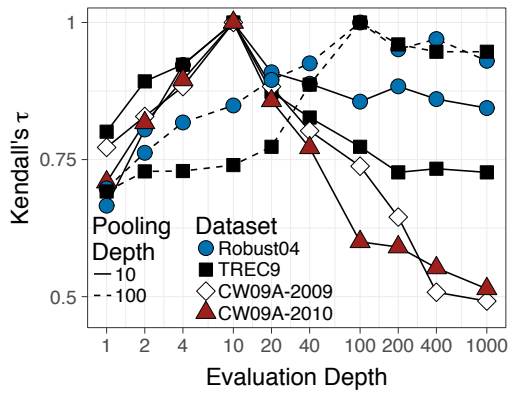
(b) Discrimination ratio with $p = 0.05$.



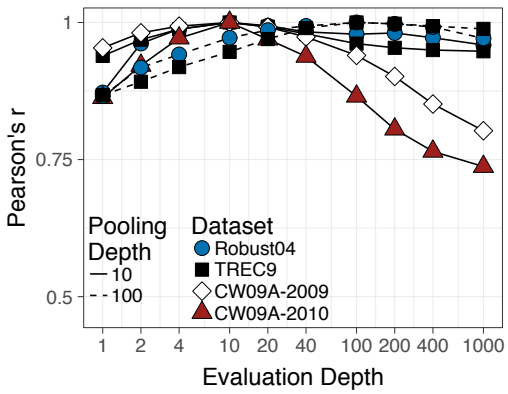
(c) Coverage ratio with $p = 0.05$.



(d) Inversion ratio with $p = 0.05$.



(e) Kendall's τ rank correlation.



(f) Pearson's correlation coefficient.

Figure 5.6: Comparisons of $\text{NDCG}_a@k$, with different datasets and pooling depth settings.

5.5.1 Metric Comparisons with Complete Judgments

Comparisons using Significance Testing. In Figure 5.6(a), at each evaluation depth k the two shallow-pooled CW09A datasets have higher median p-values for system-versus-system comparisons when compared to the Robust04 and TREC-9 test collections. This difference remains evident even when the latter two are shallow-pooled to $d = 10$. In general, the median p-value decreases as the evaluation depth increases towards the pooling depth d, as more information is introduced into the computation of metric scores. Nevertheless, note also that in the two CW09A environments the median p-value then *increases* past $k = 10$, that is, once the region of extended evaluation is reached. In Figure 5.6(b), the same relativities occur when the metric is assessed via the discrimination ratio that is carried out at significance level $p = 0.05$, rather than by median p-value. This graph supports the earlier hypothesis that a high median p value in Figure 5.6(a) is indicative of a low discrimination ratio. In particular, when $k > d$ and the evaluation depth is greater than the pooling depth, extended evaluation results in a smaller ratio of system-versus-system pairs as being deemed significant at $p = 0.05$. That is, graphs in the top row of Figure 5.6 reinforce each other in this regard.

Coverage and Inversion Ratios. The coverage ratios are plotted as a function of evaluation depth k in Figure 5.6(c). For each set of lines, a reference metric is set and according to the judgments in the original datasets, the pooling depths $d = 10$ and $d = 100$ are employed. The same pattern of behavior is again evident: as k increases towards d, coverage increases, and the at-k evaluation converges towards the at-d coverage measurements. However, once k is greater than d, the two CW09A test collections give rise to divergent outcomes, and the coverage ratio decreases. Inversion ratios are plotted in Figure 5.6(d) and it can be seen that the divergence in coverage ratio beyond $k = d$ is not merely a matter of losing the ability to derive statistical significance. The relatively high inversion ratios that arise for the two CW09A collections are evidence that a non-trivial fraction of the significant system-versus-system outcomes identified at $k = d$ are actively contradicted at larger values of k. Note also that the two earlier TREC collections, even when shallow-pooled to $d = 10$, do not behave in this manner, and do not give rise to the same high levels of inversions. This observation is consistent with the raw system scores shown in Figure 5.5.

Correlation Coefficient. The last two figures show that there is also a substantial decrease in measured system order correlation, using both Kendall's τ and Pearson's r when extended evaluation is used with the two ClueWeb collections. In these two graphs the reference metric is again taken to be NDCG$_a$@k, where d is either 10 or 100.

Discussion. Overall, all of the six depicted indicators agree that extended evaluation using the two CW09A collections gives rise to system comparisons that diverge from those that arise when the evaluation depth k is restricted to the pooling depth d. Similar behavior (not shown) was also found when the same experiments were carried out using the later CW09A-2011 and CW09A-2012 test collections.

Table 5.5 shows the discriminative power, coverage and inversion ratios of metrics other than NDCG$_a$@k. All of the comparisons are performed at significance level $p = 0.05$. In Table 5.5(a), the Robust04 dataset is used, with pooling to depth $d = 100$, and the reference depth for coverage and inversion ratio is $d = 100$. With this collection, systems are relatively stable in extended evaluation, and for $k > 100$ the coverage ratios are close to 100, while inversion ratios are very close to zero. Table 5.5(a) also confirms previous findings regarding discrimination ratio: on this dataset, NDCG$_a$@k and AP$_b$@k both have discrimination ratios of around 65%, which is similar to RBP(0.95)@k but more than ten points higher than RBP(0.8)@k. Not surprisingly, ERR@k is even less discriminative than RBP(0.8)@k.

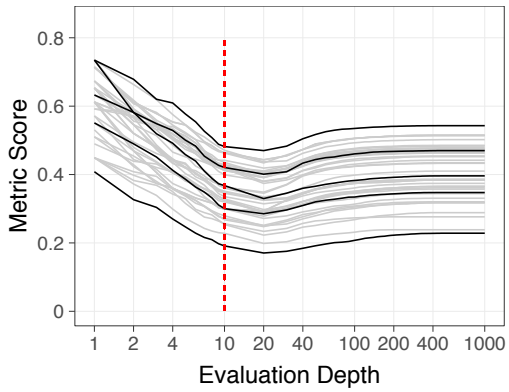
The situation is different when the CW09A-2010 dataset is used (Table 5.5(b)). On this collection, the recall-based metrics AP$_b$@k and QM@k give rise to coverage ratios well below 100% in extended evaluation, and inversion rates that are greater than 5%. On the other hand, the two RBP variants are relatively unaffected by the exact choice of k, and give rise to system-versus-system comparisons on the CW09A datasets that are resilient to the evaluation depth k. Note also the discrimination ratios in Table 5.5(b) of recall-based metrics are not higher than weighted-precision metrics as observed in Table 5.5(a). Now the strongly top-weighted RBP(0.8) metric yields system-versus-system comparisons that match or are better than the corresponding ratios for the two recall-based metrics listed.

5.5.2 Metric Comparisons with Incomplete Judgments

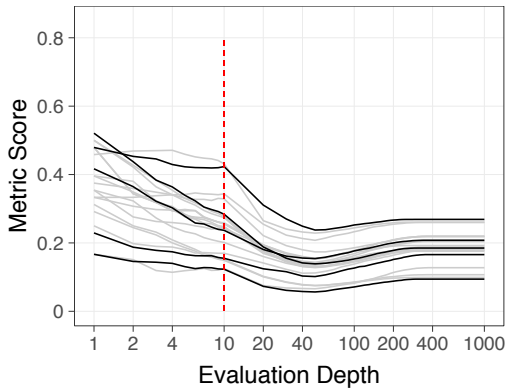
Missing Judgments. So far, the effect of using incomplete judgments in the evaluation process has been ignored. Several alternative approaches have been proposed for incomplete judgment sets. Table 5.5 suggests that it may be injudicious to apply extended evaluation to the CW09A-2010 dataset without seeking to address the problem of missing judgments.

There are various strategies to evaluate system effectiveness over partial judgments, among which two of the most widely used are BPref [15] and the condensing technique [95]. Both methods can be plugged into an existing TREC "top-d" pooling test collection. The difference is that the condensing process happens on the ranked list so that all metrics discussed above can be applied without modification, while BPref is a metric tailored to the situation that unjudged documents appear in ranked lists. Sakai [95] showed that the condensing technique actually provides competitive results to BPref. As most of the concerns result from using recall-based metrics, the condensing technique is employed here.

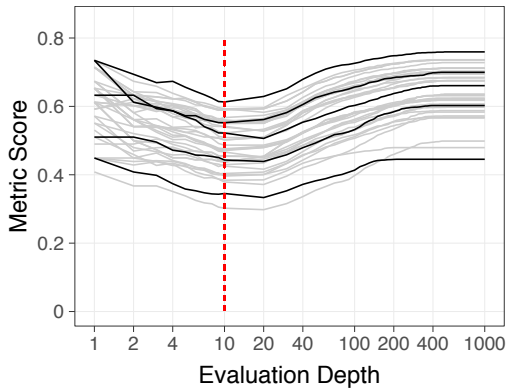
Figure 5.7 shows raw scores of two recall-based metrics AP$_b$@k and NDCG$_a$@k on collections Robust04 and CW09A-2010. Both collections are pooled to a shallow depth $d = 10$ and except for the condensing process applied to the runs, all the other settings are same as Figure 5.5. The experimental results show that, evaluating over condensed lists may help to some extent, especially when an extremely deep evaluation depths are considered. However, the cross-overs still exist regardless whether the evaluation depth exceeds the pooling depth. The same set of indicators are computed on CW09A-2010, with the application of the condensing technique. Results are listed in Table 5.6. By definition, all unjudged documents were removed from the contributing runs, and they were then processed to the shown depths, or to a depth equal to the number of



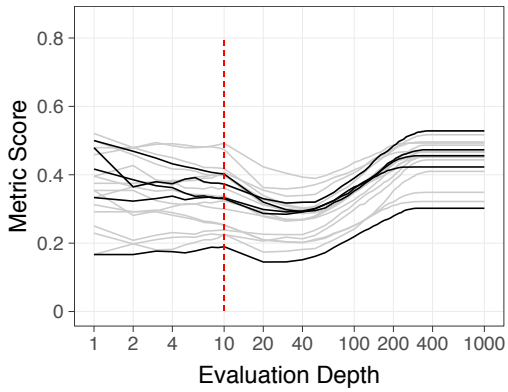
(a) Condensed $AP_b@k$ on Robust04.



(b) Condensed $AP_b@k$ on CW09A-2010.



(c) Condensed $NDCG_a@k$ on Robust04.



(d) Condensed $NDCG_a@k$ on CW09A-2010.

Figure 5.7: Condensed $AP_b@k$ and $NDCG_a@k$ scores at different evaluation depths. Both Robust04 and CW09A-2010 are pooled to $d = 10$. These can be compared with Figure 5.5(a)–(d).

judged documents in the run if that number was less than k. For the CW09A-2010 collection, that process resulted in an average per-system per-topic run length of 244.9 documents, and a maximum run length across the systems and topics of 391 documents. In terms of discrimination, coverage, and inversion ratio, there is no benefit gained by condensing the runs; if anything, condensing the runs has amplified the way in which system comparisons are affected by different k. In order to gain a thorough understanding of the extent to which the unjudged documents impact evaluation, residual information of weighed-precision metrics are calculated using Equation 2.23 and present in Table 5.7.

As noted before, weighted-precision metrics such as RBP use pre-determined weights to compute an overall score for a ranked list, and it is thus straightforward to accumulate the weights associated with any unjudged documents as a *residual* and provide an exact limit on the extent of the score uncertainty attributable to incomplete judgments. Table 5.7 shows residuals for the RBP metric with two different persistence values. As can be seen, shallow evaluations yield both higher numeric RBP scores and lower numeric residuals, implying more accuracy in the values that are presented as being "RBP scores". Note also that as k increases, the residual cannot increase; this is a useful attribute of utility-based metrics, and explains the smooth score convergence already

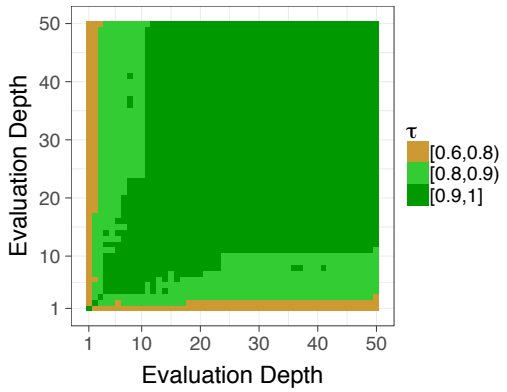
k	$AP_b@k$			$QM@k$			$RBP(0.95)@k$			$RBP(0.8)@k$		
	Disc.	Covr.	Invr.	Disc.	Covr.	Invr.	Disc.	Covr.	Invr.	Disc.	Covr.	Invr.
1	30.0	47.4	14.9	**35.7**	45.5	17.4	30.0	54.7	4.7	30.0	**61.4**	**0.0**
4	45.7	75.4	**0.0**	**48.1**	71.1	**0.0**	41.4	72.6	**0.0**	40.5	**80.2**	**0.0**
10	**52.4**	92.1	**0.0**	51.9	87.6	**0.0**	49.0	72.6	**0.0**	44.3	**96.9**	**0.0**
40	53.3	95.6	**0.0**	57.6	96.7	**0.0**	48.1	91.5	**0.0**	46.2	**100.0**	**0.0**
100	**53.3**	82.4	**0.0**	**53.3**	81.0	**0.0**	48.6	90.6	**0.0**	46.2	**100.0**	**0.0**
400	41.4	44.7	15.8	47.6	43.0	28.9	**48.6**	90.6	**0.0**	46.2	**100.0**	**0.0**

Table 5.6: Discrimination, coverage, and observed inversion ratios (all expressed as percentages at $p = 0.05$) for condensed runs, using the Robust04 test collection. The reference metric is the corresponding metric evaluated at $k = d = 100$. These values can be directly compared with the first block of Table 5.5(a)

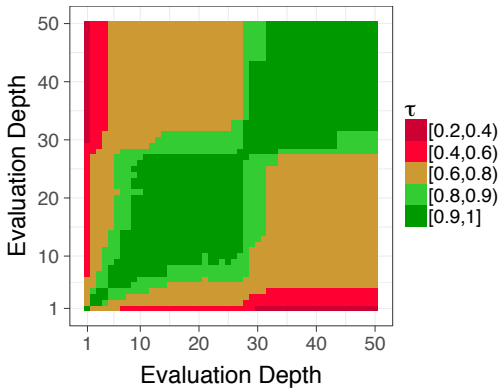
k	Robust04		
	$RBP(0.95)@k$	$RBP(0.8)@k$	$ERR@k$
1	$0.031 + 0.950$	$0.123 + 0.800$	$0.290 +$ **0.291**
4	$0.097 + 0.815$	$0.315 + 0.410$	$0.404 +$ **0.060**
10	$0.183 + 0.599$	$0.439 + 0.107$	$0.430 + 0.019$
40	$0.300 + 0.129$	$0.470 +$ **0.000**	$0.437 + 0.003$
100	$0.317 + 0.006$	$0.469 +$ **0.000**	$0.437 + 0.001$
400	$0.317 + 0.002$	$0.470 +$ **0.000**	$0.439 + 0.001$
1000	$0.317 + 0.002$	$0.470 +$ **0.000**	$0.439 + 0.001$
k	CW09A-2010		
	$RBP(0.95)@k$	$RBP(0.8)@k$	$ERR@k$
1	$0.019 + 0.950$	$0.074 + 0.800$	$0.088 +$ **0.385**
4	$0.063 + 0.815$	$0.203 + 0.410$	$0.146 +$ **0.088**
10	$0.133 + 0.599$	$0.301 + 0.107$	$0.170 + 0.044$
40	$0.249 + 0.234$	$0.332 +$ **0.005**	$0.183 + 0.020$
100	$0.265 + 0.190$	$0.332 +$ **0.005**	$0.185 + 0.020$
400	$0.266 + 0.189$	$0.332 +$ **0.005**	$0.185 + 0.020$
1000	$0.266 + 0.189$	$0.332 +$ **0.005**	$0.185 + 0.020$

Table 5.7: Run scores and residuals generated by RBP and ERR, averaged across contributing systems and across topics, for two collections, two values of the RBP parameter p, and a range of evaluation depths k. In the case of the Robust04, pooling is to depth $d = 100$; in the case of the ClueWeb09A-2010 data, pooling is to depth $d = 20$.

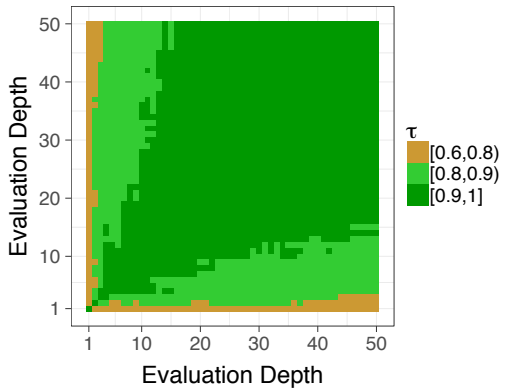
documented in Figure 5.4. Also, it can be observed that ERR behaves similar to RBP with a small p-value. While it is not possible to compute residuals for recall-based metrics, all of AP, NDCG, and QM are likely to be subject to score uncertainties to at least the same extent as $RBP(0.95)$.



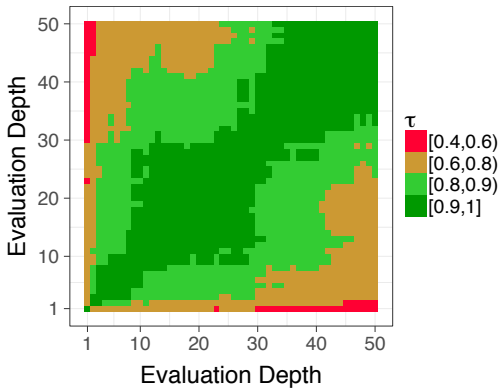
(a) $AP_b@k$ on Robust04.



(b) $AP_b@k$ on CW09A-2010.



(b) $NDCG_a@k$ on Robust04.



(b) $NDCG_a@k$ on CW09A-2010.

Figure 5.8: Recall-based metrics applied to the Robust04 and CW09A-2010 test collections, using evaluation depths $1 \leq k \leq 50$, and with the resultant system orderings compared using Kendall's τ. Both collections are pooled to depth $d = 20$. The top row shows $AP_b@k$, and lower row $NDCG_a@k$, with Robust04 on the left and CW09A-2010 on the right.

5.6 Meta-Evaluation with Different References

In Section 5.5, the pooling depth is fixed in an experiment and the system ranked list at $k = d$ is always regarded as the reference metrics in the comparisons, reflecting an experimental setting that assumes a reliable conclusion can be drawn when evaluation depth $k = d$. This section shows results of the first extension, which varies the reference point. The evaluation configurations can vary in two aspects. The first is to regard different evaluation depths of the current metric as different references and the second is to take another metric at different evaluation depths to be the reference point. The volatility matrices defined in Section 5.2.2 are computed in order to complete the two tasks.

5.6.1 Different Evaluation Depths as Reference Points

Tables 5.5 and 5.6 demonstrated that when the evaluation depth is greater than the pooling depth, system-versus-system comparisons can lead to notably different outcomes when compared to the

same evaluation carried out at the same pooling depth. If the evaluation depth is limited to be less than or equal to the pooling depth, system orderings still vary with recall-based metrics, as illustrated by the crossing score lines in Figures 5.2 (on Page 114) and 5.3 (on Page 116). The computed volatility matrices of recall-based metrics, which is defined in Equation 5.5, are visualized in Figure 5.8. In the experiments, simulated judgment sets with pooling depth $d = 20$ on both Robust04 and CW09A-2010 datasets are used. The evaluation depth is limited to $1 \leq k \leq 50$. All experiments conducted in this section no longer consider a single reference point. Instead, for each distinct value of k, a system ordering is generated using the system scores when averaged over the topic set. Kendall's τ coefficients are then computed between all pairs of system orderings, to create a volatility matrix. Finally, color coding is used to indicate different ranges for τ. If the system ordering is unaffected by evaluation depth, then all entries in the volatility matrix will be dark green. The diagonal line of each graph must be dark green; variance away from that color in other locations of the plot indicates disagreement in the induced system ordering. Correlations of 0.9 and greater represent system orderings that are very close to each other.

The four panes in Figure 5.8 correspond to two different recall-based metrics ($\text{AP}_b@k$ and $\text{NDCG}_a@k$) and two different datasets. On Robust04 (the left column) the outcome is as might be expected – the system orderings generated by the metric are largely unaffected by evaluation depth, with the only exception being that very small values of k give system orderings that compare poorly with the orderings generated by larger values of k. These results agree with the findings of Webber et al. [123], who noted that the extended evaluation gave rise to increased discrimination between systems. Markedly different behavior is observed for the CW09A-2010 (the right column). Now there are many low correlation scores in evidence, and only when the evaluation depth k is approximately the same as the pooling depth d are the generated orderings in broad agreement. Similar patterns of behavior also arise when Kendall's τ is replaced by other correlation coefficient, such as τ_{ap} or RBO.

Figure 5.9 shows the results of weighted-precision metrics on CW09A-2010. Kendall's τ correlation is still used in computing the volatility matrices for metric $\text{RBP}(0.95)@k$ (the first figure) and $\text{ERR}@k$ (the second figure). Despite the relatively low ability of $\text{ERR}@k$ to provide discrimination in system-versus-system evaluations, the figure demonstrates that it provides consistent system orderings across a broad range of evaluation depths. Also, when a persistence parameter $p = 0.8$ or less is set in RBP, a similar pattern to $\text{ERR}@k$ can be observed. The left figure also indicates that $\text{RBP}(0.95)@k$ can provide consistent system orderings across a wide range of evaluation depths and with a higher discrimination ratio (Table 5.5). This is a consequence of its gradated emphasis on documents beyond the pooling depth, with contributions arising from relevant documents right through until the evaluation depth k. At an evaluation depth of $k = 40$ the residuals are also relatively small (Table 5.7), with the exception of CW09A-2010 and $\text{RBP}(0.95)@k$. Here, the residuals are uncomfortably high when compared with the score, even at $k = 1{,}000$.

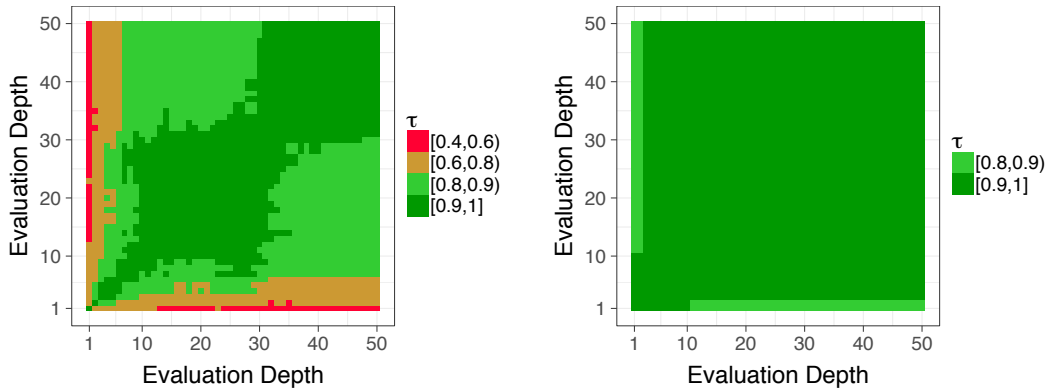


Figure 5.9: Kendall's τ of RBP(0.95) (left-hand) and ERR (right-hand) on the CW09A-2010 test collection, with $d = 20$. Both figures consider an evaluation depth $1 \leq k \leq 50$.

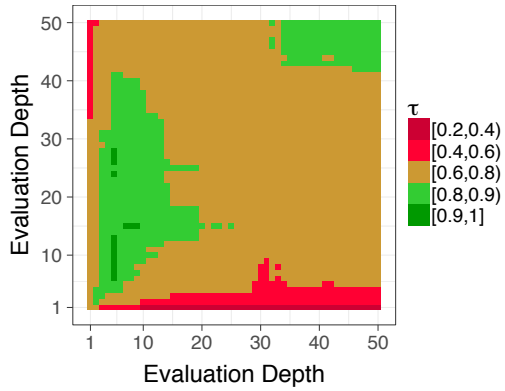
5.6.2 Different Metrics as Reference Points

As shown in the previous section, metrics behave in different ways when varying the evaluation depth. However, how well those metrics are correlated with each other remains unclear. Comparisons between system orderings and metric choice are performed in this section in order to explore the correlations between different evaluation metrics.

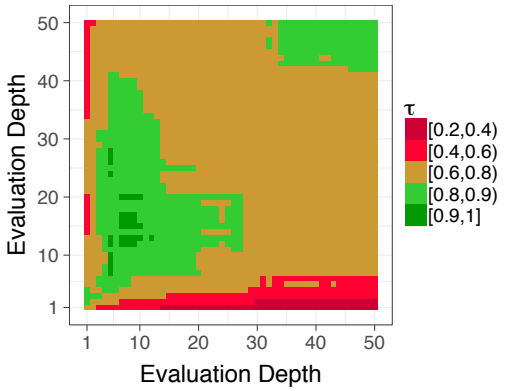
It is also possible to compare the relative behavior of metrics, to determine configurations in which they do and do not yield similar system rankings, by placing different metrics on the two axes. Figure 5.10 shows a number of such comparisons. In the first row, the normalized AP$_b$@k metric is compared with the two other truncated AP@k variants introduced in Section 2.4.5. The second row compares NDCG$_a$@k with a variant that uses an alternative Zipfian discount function, $W_{\text{M}}(i) = 1/i$ rather than $W_{\text{M}}(i) = 1/\log_2(1+i)$ [51]; and with AP$_b$@k. The third row completes the comparison by comparing RBP(0.95)@k with NDCG$_a$@k and AP$_b$@k. Several other comparisons such as NDCG$_a$@k and NDCG$_a$@k were carried out, and similar trends were observed. For example the NDCG$_a$@k and NDCG$_a$@k comparison graphs looks quite similar to the RBP(0.95)@k and NDCG$_a$@k comparison. In Figure 5.10, evaluation of NDCG using a small value of k and the logarithmic discount function behaves somewhat akin to a deeper evaluation using an alternative Zipfian discount function, perhaps because the Zipfian discount means that any gains from deep-ranked items are rendered inconsequential. In Figure 5.10(e), the strong symmetric pattern of correlation indicates that RBP(0.95) and NDCG behave similarly across a broad range of retrieval depths, despite their different discount and normalization regimes.

5.6.3 Metric Parameters Revisited

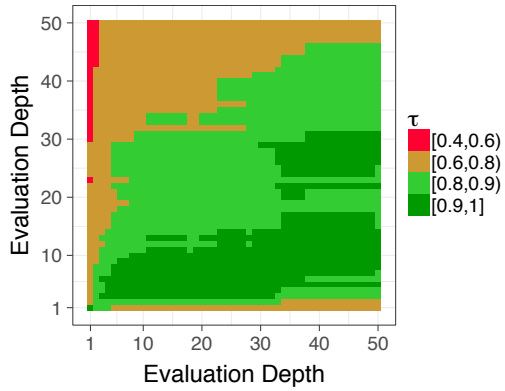
The different patterns of behavior that are evident in Figures 5.8 and 5.9 are partly a consequence of the distinct nature of the NewsWire and ClueWeb datasets (see Figure 5.1), and partly a consequence of the fact that in AP@k and NDCG@k there are two different concepts combined into a single variable. The first concept is essentially a top-weightedness parameter, corresponding to



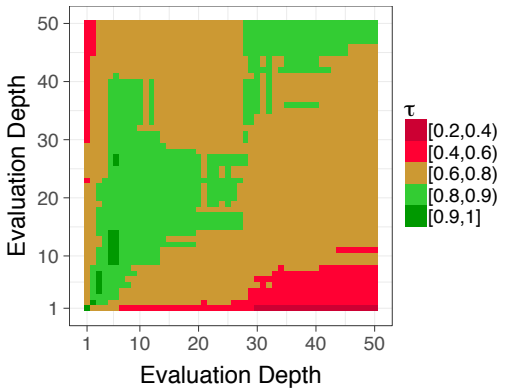
(a) $\text{AP}_a@k$ (vert.) vs. $\text{AP}_b@k$ (hori.)



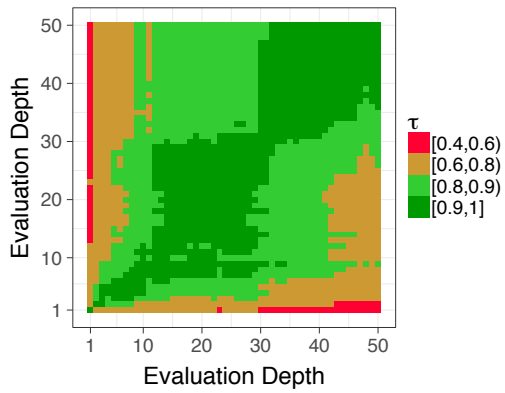
(b) $\text{AP}_c@k$ (vert.) vs. $\text{AP}_b@k$ (hori.)



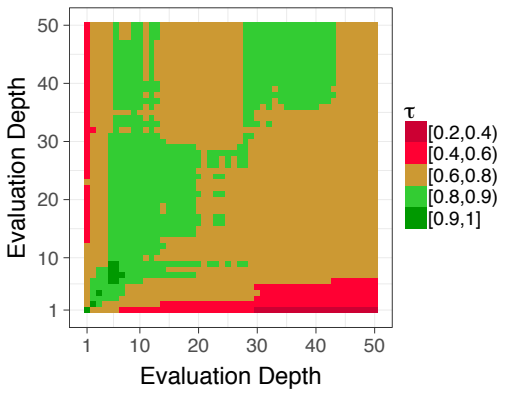
(c) $\text{NDCG}_a@k$: Log (vert.) vs. Zipf (hori.)



(d) $\text{NDCG}_a@k$ (vert.) vs. $\text{AP}_b@k$ (hori.)



(e) $\text{RBP}(0.95)@k$ (vert.) vs. $\text{NDCG}_a@k$



(f) $\text{RBP}(0.95)@k$ (vert.) vs. $\text{AP}_b@k$

Figure 5.10: Kendall's τ scores on system rankings using CW09A-2010, pooled to depth $d = 20$, including using variants of AP$@k$ and NDCG$@k$. The first row compares the normalized $\text{AP}_b@k$ metric used through the balance of this paper with $\text{AP}_a@k$ (the `trec_eval` variant), and with $\text{AP}_c@k$ (see Section 2.4.3 on Page 39). The second row compares $\text{NDCG}_a@k$ with a variant of NDCG that uses an alternative Zipfian discount strategy, and with $\text{AP}_b@k$; and the third row compares $\text{RBP}(0.95)@k$ with $\text{NDCG}_a@k$, and with $\text{AP}_b@k$.

the variable p that is used in RBP. It determines the relative weight given to each document when the relevance scores are combined to obtain an overall score for a run. For example, assuming that there are $R_d \geq k$ relevant documents and that NDCG@k is being used, the first document in the ranking accounts for $1/(\sum_{i=1}^{k} 1/\log_2(1+i))$ of the total metric value. When $k = 5$, that is around 0.34; when $k = 10$, around 0.22, and when $k = 20$, around 0.14. Elements beyond depth k are always assigned a weight of zero, in the same way that elements beyond rank k are assigned a weight of zero by Prec@k. In this context, NDCG@k_1 and NDCG@k_2 must be regarded as being different metrics, in the same way that Prec@k_1 and Prec@k_2 are, or that RBP(p_1) and RBP(p_2) are. When k_1 is close to k_2 (or p_1 close to p_2), the numeric scores and numeric behaviors of NDCG@k_1 and NDCG@k_2 (or RBP(p_1) and RBP(p_2)) are likely to be correlated. But when k_1 and k_2 are not close, there is no more requirement that NDCG@k_1 and NDCG@k_2 be correlated than there is that (say) ERR and AP should be correlated. A similar argument applies to NDCG@k. That is, the truncated metrics AP@k and NDCG@k should not be regarded as approximations of full-depth AP and full-depth NDCG; rather, they must be considered to be independent metrics in their own right. The second concept associated with k is that it determines the degree to which what is reported as the metric's score is an approximation of the correct value. For example, in the case of RBP(p)@k, the parameter p indicates the strength of top-weightedness, and the value of k then controls the extent to which the sum that is computed is an approximation of its correct value. The balance is taken to be the residual. In the case of NDCG$_a$@k and AP$_b$@k, the conflation of the two concepts – with k serving both as a top-weightedness parameter, and also a limit governing the summation – means that there is no sense of increasing k to get a "better" approximation of the metric. Increasing k changes the metric to make it less top-focused, and in doing so shifts weight further down the ranking; perversely, that shift may then make the approximation that is computed *less* accurate, since a greater fraction of the metric's weighting might consist of unjudged documents.

With AP$_a$@k, the situation is different again. Because R_d is assumed to be fixed and independent of k, AP$_a$@k is non-decreasing in k, albeit with a possibly very large residual. With this metric, it is increasing the pooling depth rather than the evaluation depth that makes it less top-weighted. That is, with AP$_a$@k the pooling depth that is used must be considered to be a parameter of the metric, rather than the evaluation depth k. A similar complex relationship exists with NDCG$_a$@k. when $k \leq R_d$ comprehensive evaluation makes the scores less top-weighted and hence risks increasing the amount of uncertainty that is implicit in the measured scores, and in downstream uses such as system-versus-system comparisons.

With weighted-precision metrics such as ERR and RBP, increasing the evaluation depth k and/or the pooling depth d serve only to reduce the uncertainty in the measured scores, and neither change can alter the degree of top-weightedness in the evaluation.

5.7 Conclusions

In this chapter, we explored the role that the metric evaluation depth k plays in determining metric values and system-versus-system evaluations, paying particular attention to the different char-

acteristics between recall-based and weighted-precision metrics. Two collection types, TREC NewsWire data and the ClueWeb dataset have been used, together with the system runs that contributed to the relevance judgments when those collections were formed. We empirically showed that pooling depth is an important factor when choosing an evaluation configuration. Even on the same test collection, the configuration of a different pooling depth and a different evaluation metric may lead to different conclusions when comparing systems.

Specifically, weighted-precision and recall-based metrics are examined separately, in addition the measurement of how well they correlated to one another. We observe that, evaluating recall-based metrics to a given depth k based on pooling to a different depth d can in some circumstances be equivalent to setting a parameter that determines the degree of top-weightedness in the evaluation, with larger values of k decreasing the weight assigned to items that appear early in the ranking. This implies that recall-based metrics should regard k as an additional parameter that when changed gives rise to a different metric; and there is no indication that increasing the evaluation depth will make more convincing conclusions with regard to the system comparisons. Indeed, with variants of two standard recall-based metrics, system-versus-system evaluations using the ClueWeb resources have been shown – in a range of different ways – to be highly dependent on the value of k that is used.

As noted before, recall-based and weighted-precision metrics are different from each other. Our experiments in this chapter further confirmed that observation. The top-weightedness parameter (p in the case of RBP, and the chosen gain regime in the case of ERR) are relatively stable in their behavior as k becomes larger than the evaluation depth. With these metrics, there is an explicit parameter that specifies the decay in weight as the set of documents comprising the ranking are one-by-one incorporated in to the computed metric value. This makes the scores that these metrics generate more resilient to changes in k and/or d, and hence makes comparisons based on them more robust as dataset resources are amended and augmented.

Results shown in this chapter re-emphasize the importance of choosing the proper configuration according to different test collections. Unless we are confident in the distribution of relevant documents, a standard metric evaluated to the standard depth should be employed. Even though a previous study [123] showed that an extended evaluation may be acceptable, it should not be regarded as being generalizable across of the test collections. Moreover, given the large impact of the evaluation depth when recall-based metrics are used, the evaluation depth should be fixed prior to the evaluation, and it should not be changed during the process. As we showed in the last section using volatility matrices, recall-based and weighted-precision metrics may agree on some comparison, but may also measure different aspects, hence have different conclusions. Instead of providing evaluation results measured using just one metric, it seems necessary to also provide conclusions from another.

We can observe from the various experiments performed in this chapter, when a deep metric is applied on a shallow pooled test collection, the system performance may be biased due to the large amount of unlabeled documents. These unjudged documents may contain a large proportion of relevant documents and even if a weighted-precision metric is adopted, unless it has been configured for shallow evaluations, it may not correctly measure the system's performance. Chapter 6

considers the problem of how can we approximate system performance in the situation where a shallow pooling depth is used but a deep effectiveness metric is employed.

CHAPTER 6

Estimating Deep Metric Scores Using Shallow Judgment Pools

Chapter 5 examined different evaluation settings by considering permutations of metrics, evaluation depths and pooling depths, showing that the evaluation depth plays an important role in the process, especially when deep metrics are applied. When using a fixed number of topics (often 50), evaluation settings which include a deep metric, a shallow judgment pool, and a deep evaluation depth, often result in less reliable evaluation results, when compared to using a deep judgment pool. One reason this may lead to bias is because the shallow judgment pool fails to identify a majority of the relevant documents, as shown in Figure 5.1. Although we suggest obtaining a reliable evaluation result by carefully selecting the evaluation depth and evaluation metric when using a shallow judgment pool, there are also cases in which a deep evaluation metric must be applied, and only a shallow judgment pool is available. A shallow judgment pool may result in two different types of bias: pooling depth bias [97] and system bias [96]. Since documents located at top ranked positions may be easier to retrieve for all systems, and relevant documents that are placed at deeper ranks may be difficult to retrieve for all systems, computing a deep evaluation metric using a shallow judgment pool may not reflect true system performance.

In this chapter, we mainly consider the pooling depth bias, which is caused by using a configuration of deep evaluation metrics and shallow judgment pools in the evaluation process. To estimate the system performance under this setting, we propose two approaches. The simplest approach is to estimate the expected gain at each rank, and use this estimate to compute the effectiveness score of the system when an unjudged document appears. We term this method the *rank-level estimator*. However, from a document-level point of view, this method fails to give a global estimation for each document, since it ignores the fact that a document may be given different ranks by different retrieval systems. To compensate for this, a second approach is introduced in this chapter, which is built on top of the rank-level estimator. We call this the *two-stage estimator*, which gives a global estimate for each document by considering one rank-level estimator in the first stage, and combines outputs from multiple rank-level estimators in the second stage. So far, we have discussed the evaluation process in a batch evaluation setting. It is also known that

Rank:	1	2	3	4	5	6	7	8	9	10	...	20	...	100	...	200	...	1000
S	1	1	1	0	0	0	0	1	0	1	...	1	...	1	...	?	...	?
S'	1	1	?	0	0	?	0	1	?	?	...	1	...	1	...	?	...	?
S''	1	1	1	0	0	0	0	1	0	1	...	?	...	?	...	?	...	?

Table 6.1: A running example of two types of bias when using a shallow judgment pool. The "zero" and "one" represent relevance values at the corresponding ranks and a "?" indicates an unjudged document. Assume S is a contributing system pooled to depth 100; S' shows the relevance values when S has been left-out; S'' shows the relevance lists when S is pooled to 10 only.

different topics may have a different number of relevant documents, which implies that a shallow judgment pool may be sufficient to obtain reliable evaluation results if a topic only has a limited number of relevant documents. Therefore, we also investigate a dynamic score estimation process, utilizing the *coefficient of covariance* (γ) [22]. That is, this chapter introduces a framework for estimating deep evaluation metrics using shallow judgment pools.

We begin in Section 6.1 by discussing different types of bias resulting from using shallow judgments, covering existing techniques that dealt with the bias in Section 6.1.2; followed by the description of selected test collections in Section 6.1.4. Section 6.2 details our proposed framework for score estimation, starting with an overview and followed by two major components: (i) rank-level estimators (Section 6.2.2); and (ii) two-stage estimators (Section 6.2.3). We evaluate estimation results from rank-level estimators and two-stage estimators separately, in Section 6.3 and Section 6.4 respectively. These experiments demonstrate the applicability of the proposed methods to estimate the performance of non-contributing systems, including via analysis performed on a set of two-strata sampling based judgments.

6.1 Background

6.1.1 Types of Bias Within the Pooling Process

The biggest problem of a shallow judgment pool is that few relevant documents are labeled. Missing a large proportion of relevant documents in the judgment pool may result in biased evaluation results. Regardless of the evaluation metrics chosen, the performance of contributing systems may be overestimated and the performance of new systems may be underestimated, for the case when unjudged documents are assumed to be not relevant, since new systems may potentially contribute unlabeled relevant documents. This type of bias is often referred to as *system bias* [96]. Aside from the biased comparison between a pooled system and a new system, the phenomenon of *pooling depth bias* [97] may also exist. This is because a shallow pool disregards relevant documents located at deep ranks, for which the retrieval of relevant documents may be more difficult – a detail that is largely ignored in this evaluation setting. Consider the extended evaluation process, relevant documents at deep ranks may still affect the final effectiveness score and if we assume they are not relevant, then the system's performance will be underestimated. For example, consider Table 6.1 and DCG@10 as the evaluation metric. If all judgments are available, as shown in

the first row, the DCG@10 score of S is 2.73. Assume S' shows the case when S is left out from the contributing systems and so the score becomes 1.32, when unjudged documents are treated as non-relevant. Most unjudged documents in S' are relevant and assuming them to be not relevant underestimates the system's performance. This is an example of system bias.

Now consider S'', which is assumed to contribute to the pool, but the pooling depth is shallower ($d = 10$) than in S ($d = 100$). A relatively shallow metric such as DCG@k ($k \leq 20$), has little impact on the conclusions, since only the top ranked documents are considered during the evaluation. However, if we consider a deep evaluation metric, for example DCG@100, then S will be underestimated. Applying shallow evaluation metrics, such as ERR or RBP($p \leq 0.8$), can mitigate the pooling depth bias, but there are still cases where a deep evaluation metric is required. Hence, it is worth considering how to estimate deep effectiveness scores when only shallow judgment pools are available. System bias is more difficult to adjust when compared to the pooling depth bias, as even shallow evaluation metrics can suffer from the unreliable system comparisons.

6.1.2 Approaches for Score Estimation

One of the most widely applied approaches for debiasing is to estimate the "true" system effectiveness score for a given evaluation metric. There are two main categories for estimating system performance based on how the set of judged documents are selected.

Random Sampling. The first method is statistical inference, where the set of judged documents are sampled according to a fixed distribution; the performance of contributing systems is then estimated from the set of sampled documents. Inferred metrics fall into this category (plotted as double-border rectangles in Figure 2.7) and they differ in how the set of judged documents are sampled. For example, both InfAP and SubAP [127] are computed based on a set of documents sampled uniformly. While a uniform sampling process can help estimate system performance, it ignores the fact that retrieval systems usually rank documents with decreasing probability of relevance. In order to make a better estimation, this property was reconsidered and a stratified sampling process was later adopted by inferred metrics such as XInfAP and InfNDCG [130]. All inferred metrics adopt a sampling process that is independent of the evaluation metric used. While this sampling can be easily operationalized, it may result in higher variance when estimating system effectiveness scores. In order to reduce the estimation variance, method-dependent sampling methods were proposed. For example, Aslam et al. [5] used a joint probability distribution that is based on the definition of AP, from which statAP is computed. $\mathcal{E}$MAP [20] first showed that AP scores follow a normal distribution, and in doing so they estimated the expected AP score based on the sampled judgments.

Although all of the previous described methods can estimate system performance to some extent, they require a special sampling process beyond the TREC pooling approach. Hence, these inferred metrics cannot be applied when the set of judged documents is selected using a top-d pooling method. A uniform or stratified sampling process may be easy to operationalize, but statAP and $\mathcal{E}$MAP are difficult to implement correctly. When a set of randomly sampled judgments are used, there is no need to discuss the "pooling depth" bias, since: (i) there is no concept

of "pooling depth" in this process; and (ii) there is no reason for the truncation of inferred metrics. Therefore, inferred metrics are not amenable to the non-random sampling process.

Estimating Based on TREC Pooling. Derived from the existing TREC pooling process, various score adjustment methods have been proposed [18, 55, 83, 121], most of which are motivated by the issue of system bias [18, 55, 121]. Among all these proposed methods, the method described by Büttcher et al. [18] is metric independent, which considers the bias adjustment as a classification problem and uses the trained classifiers to predict the relevance of unjudged documents. Although the goal of methods proposed by Jayasinghe et al. [50] is to extend the judgment pool, they can also be applied to estimate system effectiveness, since the proposed approaches aim at identifying unjudged relevant documents. Three methods were proposed by Jayasinghe et al., including a Borda count method, a classification method and a hybrid approach that combines the first two. Experimental results showed that by using the proposed method, relevant documents that can only be identified through manual assessment can also be found.

In addition to previous methods that require sampling based judgmnets, and adhering to the TREC pooling process, there are also several approaches that try to mitigate system bias [55, 83, 121], most of them consider either weighted-precision or recall-based metrics, but not both. For example, approaches proposed by Webber and Park [121] and Lipani et al. [55] may be generalized to other metrics, their core idea is based on residual information computed in the weighted-precision metrics. By repeating leave-one-out experiments for all contributing systems, an adjustment factor can be obtained and applied for evaluating new retrieval systems. Also making use of residual values but not restricted to the system bias, Ravana and Moffat [83] (the RM method) considered three different approaches to estimate system performance, based on both collection information and the known performance of a system. Methods of this form differ from earlier approaches in that they can be applied with most weighted-precision metrics. Approaches proposed by Büttcher et al. [18] and Ravana and Moffat [83] can deal with both types of bias to some extent, while methods proposed by Webber and Park [121] and Lipani et al. [55] considered only the cases of system bias.

It is worthy of note that the RM method can be applied with not only the TREC pooling approach, but also a sampling based pooling strategy. Let InfRBP be the RM method applied on a two-stratum sampled set of judgments, and let J' be the initial set of judgments pooled to depth d' and J_s be the second sampled set of judgments. We computed InfRBP on two-stratified sampled judgments as:

$$\hat{\mathrm{M}}@k = \sum_{\substack{j=1 \\ s_j \in J'}}^{k} r_j \cdot W_{\mathrm{M}}(M)(j) + \lambda \cdot \sum_{\substack{j=1 \\ s_j \in J_s}}^{k} r_j \cdot W_{\mathrm{M}}(M)(j)\,, \tag{6.1}$$

where

$$\lambda = \left(\sum_{\substack{j=1 \\ s_j \notin J'}}^{k} W_{\mathrm{M}}(M)(j) \right) \cdot \left(\sum_{\substack{j=1 \\ s_j \in J_s}}^{k} W_{\mathrm{M}}(M)(j) \right)^{-1},$$

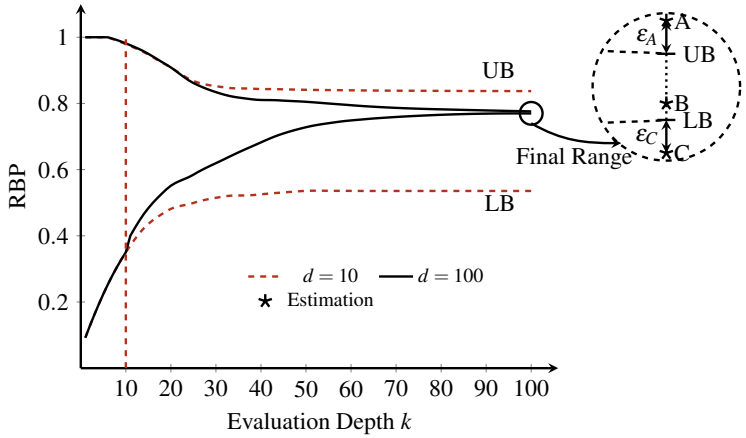


Figure 6.1: An example of an RBP score when using $d = 10$ and the $d = 100$ judgment pools. The solid line shows the case that $d = 100$ is employed and the dashed line shows the case that $d = 10$ is used. The three points "A", "B" and "C" indicate the three possible estimations using the $d = 10$ judgment pool. The "Final Range" indicates that when $k = 100$ and $d = 100$, there is still some uncertainty, but less than when evaluation is performed using the $k = 100$ and $d = 10$ evaluation configuration. "LB" is the lower bound score, which assumes unjudged documents are not relevant; and "UB" is the upper bound score, which is computed by assuming unjudged documents are relevant. The residual value $\Delta = \text{UB} - \text{LB}$.

and where the second term in Equation 6.1 estimates the total gain associated with documents contained in the second stratum. Here, λ is the interpolation estimator. Note that Equation 6.1 only adapts the RM method for sample-based judgments. A more sophisticated sampling scheme can be applied but we choose a two-stratified sampling strategy as suggested by Voorhees [117].

6.1.3 Evaluating Estimation Accuracy

There are two different aspects that must be considered when evaluating the quality of an estimation: the estimation error of the metric scores; and the correctness of system orderings prediction.

Mean Absolute Error and Root-Mean-Square Error. An elementary approach for evaluating estimation quality is to calculate the difference between the estimated performance and the "true" performance of a system. We refer this difference as mean absolute error (MAE). Similarly, a classic Root-Mean-Square Error (RMSE) can also be calculated, adding a high penalty in connection with large discrepancies. However, when metric scores are no longer represented as a point but as a range, RBP scores for example, then the best way to calculate MAE and RMSE is not so obvious. Consider the residual calculated by RBP, as defined in Equation 2.23, as an example. As shown in Figure 6.1, suppose a depth 100 judgment pool can find the majority of relevant documents for a topic, and the solid line shows the "true" effectiveness score of a retrieval system. By definition, the RBP score is represented using an interval, as shown in "Final Range". When only a set $d = 10$ judgments are available, as shown by the dashed line, the lower bound score is much smaller than the true effectiveness score (solid line). Using the set of $d = 10$ judgments, we can make estimations of the true system effectiveness and there are three possible cases, shown as

"A", "B" and "C" in Figure 6.1. The error values can be calculated using the difference between the estimated values and the two score bounds (LB, UB). Let $\hat{\text{M}}_i$ be the estimated metric M score of system i, then the absolute estimation error ϵ_i for each topic is:

$$\epsilon_i = \begin{cases} \hat{\text{M}}_i - \text{UB}, & \text{if } \hat{\text{M}}_i > \text{UB} \\ 0, & \text{if LB} \le \hat{\text{M}}_i \le \text{UB} \\ \text{LB} - \hat{\text{M}}_i & \text{if } \hat{\text{M}}_i < \text{LB}. \end{cases} \tag{6.2}$$

This definition respects the residual range, and only gives non-zero values if the estimated effectiveness falls outside the score range arising from the use of "complete judgments" at depth d. Let n be the number of systems estimated, then for each topic, RMSE is computed as:

$$\text{RMSE} = \sqrt{1/n \sum_{i=1}^{n} \epsilon_i^2}, \tag{6.3}$$

where ϵ_i is defined in Equation 6.2. The overall RMSE value is averaged over the entire set of topics for each dataset considered.

Weighted Kendall's τ distance. The second criteria for evaluating the estimation results is how well they approximate system orderings in a batch evaluation setting consisting of n systems. There are various measurements for examining the correlation between two ranked lists, as discussed in Section 2.4.5. Aside from computing the correlation coefficient, examining the distance between two lists is an alternative way of measuring the agreement between two metrics. One straightforward method is to compute Kendall's τ distance, which counts the number of inverted pairs between two n-item orderings. In IR batch evaluation where the system performance is examined using a set of topics, we consider the relative order of system S_i and S_j based on effectiveness metric means over a topic set. Let $\sigma_{i,j}$ represent the pairwise relationship between the effectiveness metric means $\bar{\text{M}}_i$ and $\bar{\text{M}}_j$ of systems S_i and S_j over a set of topics according to one measurement regime, with $\sigma_{i,j} \in \{-1, 0, 1\}$ indicating that $\bar{\text{M}}_i < \bar{\text{M}}_j$, that $\bar{\text{M}}_i = \bar{\text{M}}_j$, and that $\bar{\text{M}}_i > \bar{\text{M}}_j$, respectively. Let $\sigma'_{i,j}$ be the corresponding values for a second measurement regime and the system means that it induces, for example, using pooling to a different depth. Then Kendall's normalized τ distance is the number of pairs $1 \le i < j \le n$ in which $\sigma_{i,j} \cdot \sigma'_{i,j} < 0$, divided by $n(n-1)/2$ to bring it into the range $0 \le \tau_{dist} \le 1$, with 0 meaning "identical".

Paired t-tests are often used to quantify the strength of the relationship between two systems, and the values $\sigma_{i,j}$ and $\sigma'_{i,j}$ can also be defined as continuous values rather than discrete ones. Kumar and Vassilvitskii [53] describe a weighted τ distance that counts the strength of each discordant pair, focusing solely on cases where $\sigma_{i,j} \cdot \sigma'_{i,j} < 0$. In practice we are not only interested in the discordant pairs, but also in pairs that are deemed to be significantly different according to one of the measurement regimes but not the other, even if their overall relationship is concordant.

Suppose that $\bar{\text{M}}_i > \bar{\text{M}}_j$ according to mean effectiveness scores of the two and that a paired one-tail statistical test across topics yields $p_{i,j}$. Values of $p_{i,j}$ near zero indicate a significant

superiority of S_i over S_j; values close to 0.5 indicate that it is by chance. If we define

$$\sigma_{i,j} = \begin{cases} 0.5 - p_{i,j} & \text{if } \bar{\mathrm{M}}_i > \bar{\mathrm{M}}_j \\ 0.0 & \text{if } \bar{\mathrm{M}}_i = \bar{\mathrm{M}}_j \\ p_{j,i} - 0.5 & \text{if } \bar{\mathrm{M}}_i < \bar{\mathrm{M}}_j \,, \end{cases}$$

then $-0.5 \le \sigma_{i,j} \le 0.5$ is a real-valued quantity that captures both the direction and strength of the relationship between the two systems. We compute $\sigma'_{i,j}$ similarly by considering both direction and strength, and then, to compare the difference between the two alternative rankings of the n systems, using:

$$\mathit{dist} = \sum_{1 \le i < j \le n} \alpha \cdot |\sigma'_{i,j} - \sigma_{i,j}| \,, \tag{6.4}$$

where $\alpha \ge 0$ is an additional scaling factor. For example, if $\alpha = |\sigma_{i,j}|$ then the strength of the relationship between S_i and S_j according to the first measurement regime also influences the measured distance. Overall, if *dist* ≈ 0, then the two measurement regimes agree in terms of both the direction of each pairwise relationship S_i versus S_j, and also its strength. If *dist* is substantially greater then zero, then the two measurement regimes give rise to many system pairs for which there are non-trivial disagreements (including in both discords and in concords) over the strength of the measured relationships. Compared with Kendall's τ distance, Equation 6.4 operates over continuous values, which makes it both resistant to inconclusive changes in rank position, and also sensitive to differences in which the direction of the relationship between S_i and S_j stays the same, but the statistical strength varies markedly.

6.1.4 Experimental Materials

The collections and configuration parameters used in this chapter are shown in Table 6.2. We also measured a range of behaviors using the TREC-7 and TREC-8 collections, but do not include them. This is because these two test collections are mainly used for post-hoc analysis and also because a similar observation to earlier TREC collections can be made.

Pooling to different depths is simulated using the identified contributing systems, and the average number of judgments required per topic at different pool depths is also shown in Table 6.2, together with the corresponding percentage of documents identified as being relevant. In the experiments measuring rank stability, we also examine the two-strata sampling method described by Voorhees, and averages over ten runs for this randomized approach are included in the table. For the Robust04 task the last 49 topics are used, and judged to a depth of 100; for other tasks, we use all of the original topic sets and judgments. Our goal in collection selection was to capture as much variety as possible. The type of document collections used in each task was described in Section 2.5. The ClueWeb test collection is representative of newer collections, which are large, and have more uncertainty associated with the judgment coverage – the core issue which motivated our investigation. We show results for this dataset as a practical application of our work, noting that a pooling depth of $d = 20$ cannot provide a ground truth for a deep metric, as shown in the previous chapter. Figure 2.9 and Figure 5.1 visualized differences between the two types of

Dataset	d	$\|S\|$	Judgments per topic				2-strata
			$d' = 10$	$d' = 20$	$d' = 30$	$d' = d$	
TREC-5	100	76	272 (13)	512 (10)	747 (8)	2298 (4)	–
TREC-9	100	59	174 (11)	322 (8)	462 (7)	1382 (4)	294 (7)
TREC-10	100	54	182 (13)	335 (10)	480 (9)	1402 (5)	303 (9)
Robust04	100	42	75 (25)	139 (18)	206 (15)	710 (7)	134 (15)
TB04	80	33	164 (31)	313 (27)	453 (25)	1121 (19)	270 (25)
TB05	100	34	111 (41)	202 (36)	291 (33)	878 (25)	187 (33)
TB06	50	39	141 (31)	270 (26)	394 (23)	633 (19)	–
CW09-2010	20	21	98 (30)	–	–	187 (28)	–

Table 6.2: Datasets used: d is the original pooling depth and provides the reference point for metric scores; d' is a notional pooling depth used our experimentation; and $|S|$ is the number of contributing runs. Only Ad Hoc task runs are used. The middle four column pairs show the number of judgments averaged across topics at each pooling depth d', and the percentage of relevant documents. The last column shows the statistics when using two-strata sampling [117], averaged over topics and over ten random folds.

test collections. We can observe from both figures that the pooling depth on large web-based test collections do not cover a majority of relevant documents and a very shallow pooling depth, for example $d' = 10$, is also insufficient even for earlier TREC test collections.

We use RBP(0.95) for training and testing, as a representative for modeling a deep weighted-precision metric. Note that RBP supports the use of graded relevance, which is required since our estimations are not on a binary scale, and it also supports computing residual values providing the basis for evaluating the estimation quality. Besides, with $p = 0.95$, this metric gives similar system orderings to AP and NDCG [74]. The estimated background gain of each document is generated via training using RBP(0.95) can also be used to compute other weighted-precision measures, such as the truncated metric Scaled DCG@k when $k > d$.

6.2 Approaches for Estimating System Effectiveness for Deep Metrics

The work described in this chapter mainly focuses on the pooling depth bias rather than system bias. The challenge is to develop a method that estimates the depth-d effectiveness score of a contributing system based on a pooled-to-d' judgment set J', and minimizes the average value of ϵ_i. We will start by describing the entire estimation framework in Section 6.2.1 and then detail the two main components in Sections 6.2.2 Section 6.2.3.

6.2.1 Overview of the Estimation Framework

We first show an overview of our proposed estimation framework in Figure 6.2. The framework takes a set of judged documents as input, along with runs generated by the contributing systems. Note that we use no other ranking information than that provided by the set of contributing systems. In order to obtain final estimates for all unjudged documents, there are two major

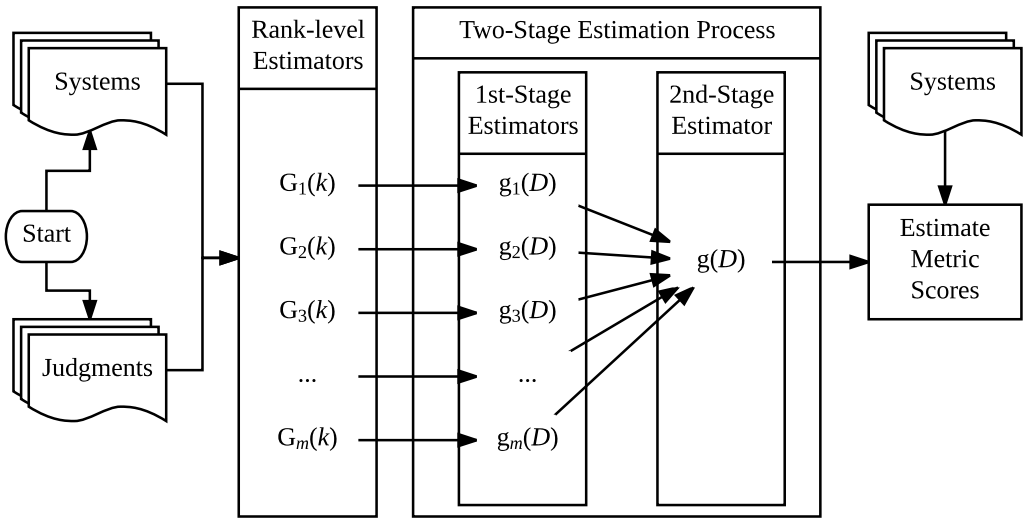


Figure 6.2: An overview of the estimation framework. The entire framework takes a set of contributing systems and the set of judgments as input, and outputs the estimated effectiveness gain for all unjudged documents.

components, shown as "Rank-level Estimators" and "Two-Stage Optimization". As is shown, the second optimization framework makes use of the output from the rank-level estimators. Within the two-stage optimization process, the second stage estimator is dependent upon the results from the previous stage to produce a final estimated effectiveness gain for the unjudged documents. We use the estimation function $g(D)$ to estimate the possible effectiveness gain of an unjudged document, and finally the effectiveness scores for the set of contributing systems. The two main components, namely, the rank-level estimators and the two-stage estimation process, are detailed in the following sections.

6.2.2 Modeling Effectiveness as a Function of Retrieval Depth

Gain Models. The core idea of designing the rank-level estimators is to model effectiveness as a function of retrieval rank. Consider a weighted-precision metric, which is computed using

$$\mathrm{M} = \sum_{i=1}^{\infty} W_{\mathrm{M}}(i) \cdot \mathit{gain}(r_i),$$

where $W_{\mathrm{M}}(i)$ is the rank-independent weight attached to the item at rank i according to the metric definition, and $\mathit{gain}(r_i)$ is the mapped relevance value at rank i. Different $\mathit{gain}(\cdot)$ modes can be applied in the metrics, as discussed in Section 2.4.2 (See Page 30). When the judgments are incomplete, and the value r_j is not known for one or more ranks j, we propose that an estimated gain $\hat{r}_j$ be used, where $\hat{r}_j$ is computed via a model of relevance in which topic and retrieval rank j are the inputs. Focusing on a single topic, we let $\langle r_{k,n} \rangle$ be a *relevance matrix* spanning n systems that have contributed to a pooled evaluation to a maximum run length (or evaluation depth) of $k = d$, so that $r_{i,s}$ is the gain attributed to system s by the document placed at rank i. The

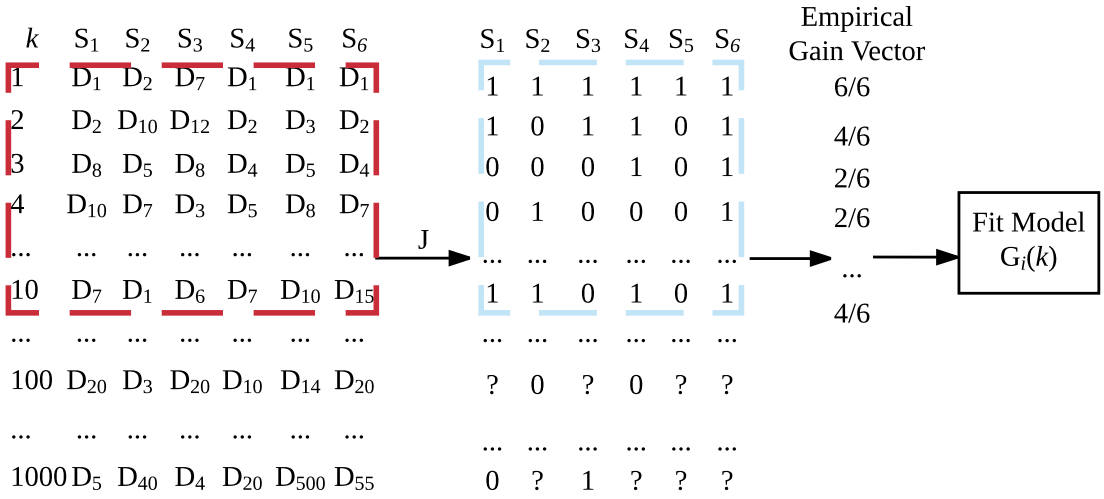


Figure 6.3: A running example for constructing rank-level estimators. The leftmost block represents the set of contributing systems; the middle block shows the relevance matrix after mapping relevance values for each document using judgment set J; the "Empirical Gain Vector" is calculated using Equation 6.5. $G_i(k)$ is a model that estimates gain at rank position k and it can be one of the models listed in Table 6.3.

empirical gain vector $\mathbf{g} = \langle g_1, g_2, \ldots, g_k \rangle$ $(1 \le k \le d)$ is then:

$$g_i = \frac{1}{n} \sum_{j=1}^{n} r_{i,j}. \tag{6.5}$$

Figure 6.3 shows the process. Suppose there are six contributing systems and the pooling depth is $d = 10$ and the binary gains are being used. After mapping each document in the contributing systems to its relevance value, we obtain the binary gain matrix shown in the middle. For each rank, we compute the empirical gain using Equation 6.5. For example, at rank position one, the empirical gain is $6/6 = 1$. The computation process repeats until the rank position $k = d$, generating a gain vector $\mathbf{g}$ of d values.

We formulate the task of estimating $\hat{r}_j$ as a regression problem. In order to estimate the relevance values where an unjudged document occurs, we propose different *gain models* to fit the change of relevance values with regard to the retrieval ranks. A gain model is a function $G(k)$ that generates a value $\hat{r}_k$ as an approximation for g_k, the empirical gain at rank k. For example, one simple gain model is to assert that if a document is unjudged its predicted gain is minimal, that is, $G_0(k) = \textit{min\_rel}$, where *min_rel* is the lower limit to the relevance range and is usually zero. This is the pessimal approach to dealing with unjudged documents that was discussed in Chapter 5. Similarly, the residuals associated with RBP combine $G_0(k)$ at one extreme, and $G_1(k) = \textit{max\_rel}$ at the other, where *max_rel* is the upper limit to the gain range, and is often (but not necessarily always) one.

Increasingly Flexible Models. We are interested in gain models that lie between the extremes of $G_0(\cdot)$ and $G_1(\cdot)$, and consider five different interpolation functions in our evaluation, embodying

Model	Description	Parameters	Assumptions
G_s	$(\textit{max\_rel} - \textit{min\_rel})/2$	–	Static, constant across all ranks
G_c	$\begin{cases}\lambda_0 & 1 \le k \le \theta \\ 0 & k > \theta\end{cases}$	$\lambda_0,\ \theta$	Constant until rank θ, zero thereafter
G_ℓ	$\max\{-\lambda_0 \cdot k + c, 0\}$	$\lambda_0 \ge 0,\ c$	Linear, decreasing until rank θ, zero thereafter
G_z	$\lambda_0/(k^c \cdot H_{n,c})$	$\lambda_0, c \ge 0$	Zipfian, monotonic decreasing, never zero
G_w	$\lambda_0 \cdot \left((1-\lambda_1)^{(k-1)^c} - (1-\lambda_1)^{k^c}\right)$	$\lambda_1 \in [0,1]$, $c > 0, \lambda_0$	Weibull, might increase before decreasing, never zero

Table 6.3: Five possible gain models, where $k \ge 1$ is the rank, and "Parameters" lists the free parameters in the estimated model.

different assumptions as to how gain varies according to rank. Table 6.3 lists the five options. The first model listed, $G_s(\cdot)$, assumes that the gain is static and both topic and rank invariant. For early ranks this is perhaps more realistic than using G_0 or G_1, but is intuitively implausible for large ranks, since the goal of any retrieval system is to bring the relevant documents to the top of the ranking.

The second model is a truncated constant model, G_c, which is predicted on the assumption that all relevant documents appear in a random manner at the early ranks of each run, and that beyond some cutoff rank m, no further relevance gain occurs. This model is rank-sensitive in a binary sense, and because m is a parameter that is selected in the context of a particular topic, it is also topic-sensitive. That is, the constant model G_c adds a level of flexibility to the static $G_s(\cdot)$, and while it may not be likely to assert that average gain is a two-valued phenomena determined by rank for any individual topic, in aggregate over a set of topics, each with a fitted value of m, the desired overall behavior might emerge.

The third step in this evolution is the model G_ℓ. The constant model G_c allows an abrupt change in predicted gain as a function of rank, at the topic-dependent cutoff value m. If we add further flexibility and suppose that average relevance gain decreases linearly as ranks increase, rather than abruptly, we get G_ℓ. This model also has cutoff rank m beyond which the expected gain from an unjudged document is presumed to be zero, given by $m = \lceil c/\lambda_0 \rceil$.

A fourth option is to allow a tapered decrease, and this is what G_z achieves, via the Zipfian distribution, in which $H_{n,c}$ is a normalizing constant determined by the controlling parameter c and the ranking length n. The expected gain rate decreases at deeper pooling depths but remains non-zero throughout, due to the long-tailed property of the Zipfian distribution. As shown in the left pane of Figure 6.4, G_z may decrease slowly, depending on the parameters used.

All four previous models have the common assumption that the effectiveness gain achieves a maxima at top ranks, and then decreases with the retrieval rank. The last model, G_w, relaxes

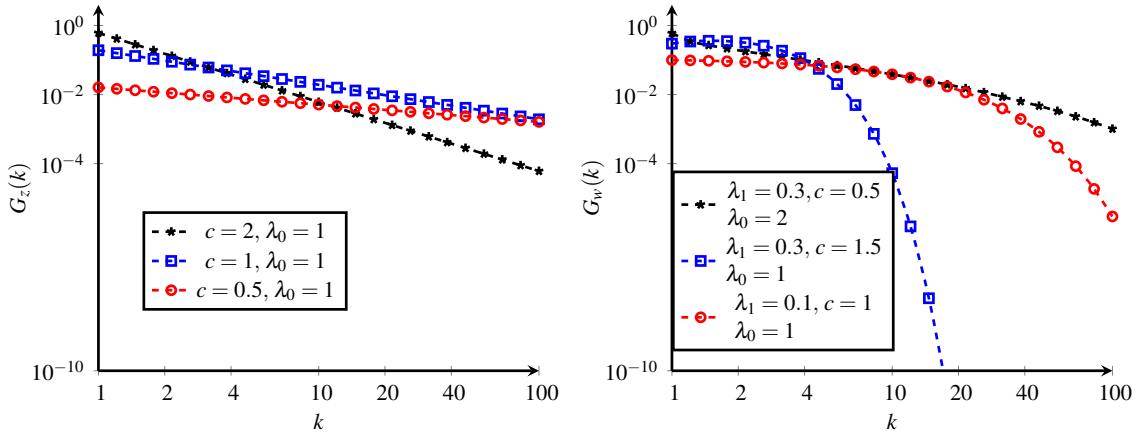


Figure 6.4: Examples of G_z and G_w, with different parameter values. The x-axis is the retrieval rank, ranging from 1 to 100, in log-scale. The y-axis is the estimated relevance at corresponding ranks. Note that both models are not zero, although the tail values are close to zero. Note that both axes are in log-scale.

this assumption, and allows the gain to initially increase or be constant, and then decrease overall. This flexibility is achieved by having more parameters, as listed in Table 6.3. In particular, when $c = 1$, the underlying distribution becomes a simple decreasing geometric distribution. We show three different configurations in Figure 6.4. The black line reflects a situation similar to the one captured by G_z, where the estimated effectiveness gain decreases as the rank increases, and there is little gain at retrieval ranks beyond 10. The red line also captures a similar trend of effectiveness change, but is different in terms of a smaller maxima and slower diminishing trend. The blue line marks the major distinction, where most systems perform the best at ranks other than the first. Since this model is derived from a discrete Weibull distribution, the gain rate decreases faster than G_z when the distribution of relevance by rank is similar.

Given a model G that has been determined in response to a empirical gain vector $\mathbf{g}$, we take $\hat{r}_j = G(j)$ for unjudged documents when r_j is unavailable, and then compute a weighted-precision metric such as RBP in exactly the same manner as before. That is, the estimated gain for that topic is used whenever the actual gain is unknown.

Goodness-of-Fit Evaluation. Regression was used to compute the two or three parameters for each model (Table 6.3), fitting them on a per-topic basis, and using a range of nominal pooling depths d. In the static model $G_s(\cdot)$ the predicted gain was set to 0.5 at all ranks; and in the constant model $G_c(\cdot)$ the cutoff parameter m was capped at the current pooling depth. Note that the large volume of input data used per topic and the small number of parameters being determined means that there is only modest risk of over-fitting, even when d is small. Predictive experiments that bypass even this low risk are described shortly. Table 6.4 lists RMSE scores on testing ranks, categorized by dataset, by model, and by pooling depth d. Looking in detail at Table 6.4, we conclude that a shallow pooling depth may lead to higher RMSE values in the extrapolation process. This can be observed across all test collections. The lowest RMSE for each "d', collection" combination in boldface. Notice that, on earlier TREC test collections TREC-

d'	TREC-5					TREC-9				
	G_s	G_c	G_ℓ	G_z	G_w	G_s	G_c	G_ℓ	G_z	G_w
10	0.3771	0.1478	0.0989	0.0929	**0.0789**	0.4081	0.112	0.0919	**0.0855**	0.0972
20	0.3838	0.1348	0.0845	0.0732	**0.0584**	0.414	0.1011	0.0702	**0.0648**	0.066
30	0.3891	0.1261	0.0774	0.061	**0.0476**	0.4182	0.0948	0.0638	0.0534	**0.0527**
40	0.3936	0.1194	0.0728	0.0544	**0.0418**	0.4218	0.0898	0.0553	0.0480	**0.0448**
50	0.3972	0.1141	0.0670	0.0497	**0.0376**	0.4249	0.0863	0.0520	0.0444	**0.0396**

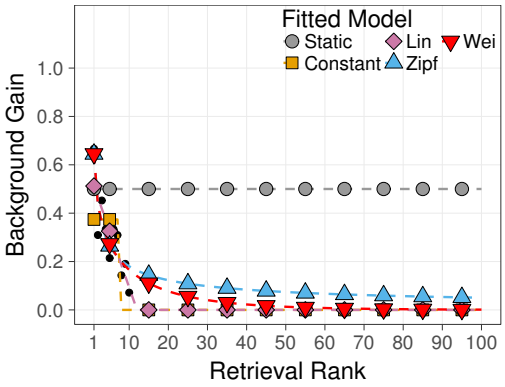
d'	TREC-10					Robust04				
	G_s	G_c	G_ℓ	G_z	G_w	G_s	G_c	G_ℓ	G_z	G_w
10	0.3894	0.1382	0.1037	0.0906	**0.0855**	0.3674	0.1825	0.1412	0.1696	**0.1227**
20	0.3958	0.1259	0.0866	0.0721	**0.0681**	0.3771	0.1598	0.1063	0.1237	**0.0823**
30	0.4010	0.1173	0.078	0.0633	**0.0600**	0.3853	0.1441	0.0944	0.1070	**0.0696**
40	0.4049	0.1116	0.0726	0.0559	**0.0504**	0.3926	0.1315	0.0864	0.0967	**0.0671**
50	0.4088	0.107	0.0667	0.0510	**0.0458**	0.3986	0.1225	0.0816	0.0878	**0.0611**

d'	TB04					TB05				
	G_s	G_c	G_ℓ	G_z	G_w	G_s	G_c	G_ℓ	G_z	G_w
10	0.2631	0.3096	0.1695	**0.1043**	0.1615	0.2345	0.3997	0.2160	**0.1299**	0.1984
20	0.2673	0.2993	0.1406	**0.0913**	0.1251	0.2371	0.3865	0.1869	**0.1125**	0.1555
30	0.2718	0.2912	0.1139	**0.0815**	0.0935	0.2387	0.3742	0.1420	**0.1066**	0.1160
40	0.2763	0.2842	0.1001	**0.0776**	0.0815	0.2418	0.3645	0.1258	**0.0992**	0.1030
50	0.2799	0.2781	0.0869	0.0736	**0.0718**	0.2442	0.3566	0.1100	0.0936	**0.0915**

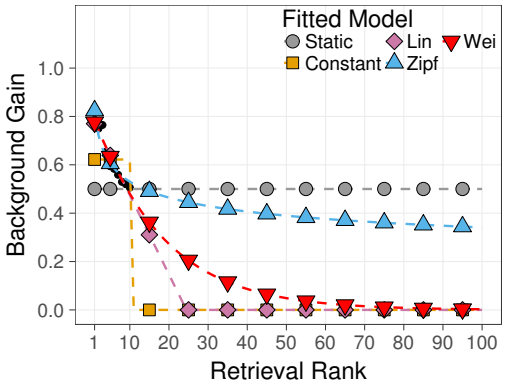
Table 6.4: The goodness-of-fit evaluation on fitting curves. Note that the RMSE values only measure how well each model describes empirical gain. A smaller RMSE value indicates a better regression curve, but not necessarily a good estimate for system performance. The bold values are the smallest RMSE computed using the current pooling depth on the corresponding test collection. RMSE values are computed on testing ranks only ($d' \le k \le d$).

5/9/10 and Robust04, G_w has the lowest RMSE most of the time, except when $d' \in \{10, 20\}$ on TREC-9. However, when considering two relatively large web test collections, TB04 and TB05, G_z is slightly better, until a $d' = 50$ pool is used. It is also apparent that little separates the Zipfian $G_z(\cdot)$ and linear $G_\ell(\cdot)$ approaches, and that either could be used as a second-choice to the Weibull mechanism. Finally in connection with Table 6.4, the consistency of values with increasing pooling depth of each test colletion confirms the earlier claim that there is only a modest risk of over-fitting affecting the results of this experiment.

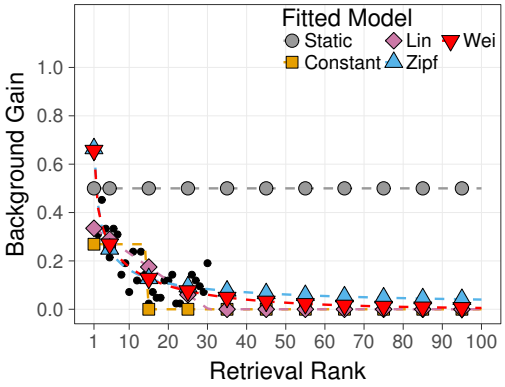
Figure 6.5 illustrates the five fitted curves for two topics, and their approximation of the empirical gain, which is shown in the graphs as a sequence of black dots. One topic from each of the two datasets is plotted, with two different pooling depths – one graph in each vertical column using all of the available judgments ($d = 100$ for Robust04, and $d = 100$ for TB05, in the bottom row), and the other two figures show fitting cases when pooling was reduced to a nominal $d' = 10$ (top row) and $d' = 30$ (middle row), respectively. One observation is immediately apparent, and that is that empirical gain does indeed decrease with rank; moreover, in the case of TB05



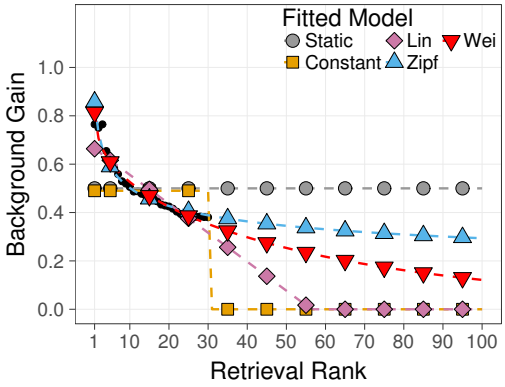
(a) Rob04, topic 673 and $d' = 10$.



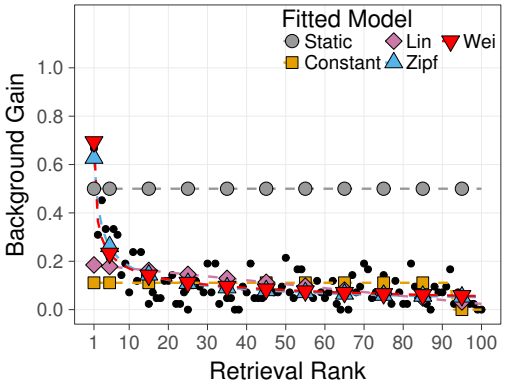
(b) TB05, topic 751 and $d' = 10$.



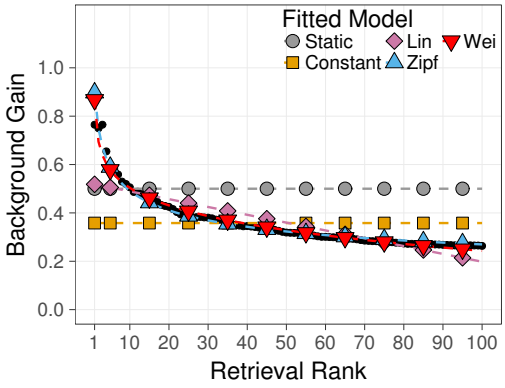
(c) Rob04, topic 673 and $d' = 30$.



(d) TB05, topic 751 and $d' = 30$.



(e) Rob04, topic 673 and $d = 100$.



(f) TB05, topic 751 and $d = 100$.

Figure 6.5: Topic 673 for Robust04 (left column), and topic 751 for TB05 (right column), with models fitted using depth $d' \in \{10, 30, 100\}$. The black dots show the empirical gain, used as basis for each set of experiments.

topic 751, it does so surprisingly smoothly. Also worth noting is that the empirical gain for the Robust04 topic decreases more quickly than it does for the TB05 topic as the evaluation depth k increases, which both fits with the overall data plotted in the right pane of Figure 2.9, and helps explain the better TB05 scores for the static model in Table 6.4. Comparing the top two graphs, it is clear that when only $d' = 10$ judgments are available, the models all diverge markedly from the actual g_k values when they are extrapolated beyond the fitted range, on both test collections. The smoother nature of the TB05 empirical gain function means that all proposed fitting models have smaller RMSE values compared to Robust04, which is reaffirmed in Table 6.4. When the full set of judgments are uesd, empirical gains on TB05 have a long-tail property, which indicates that when only shallow pooling depth is available, models such as G_e that have a heavy-tail property may incur a smaller RMSE in the extrapolated ranks. It is also worth noting that a static model with $G_s = 0.5$ throughout all ranks works better on TB05 than Robust04, which also confirms the long-tail empirical gain property.

Discussion. We proposed five different models to capture the trend of the background effectiveness gain as a function of retrieval ranks. However, there is no evidence showing that the estimation will benefit from the flexibility of any one model, thus it is difficult to tell which model can best describe the changing trend. As shown in the goodness-of-fit evaluation, almost all models have a high RMSE value when only a shallow pooling depth is used. Also, there is no model that consistently outperforms all the others across the set of test collections and for all considered pooling depths.

Since the proposed models consider background gain estimates at the rank-level, we may only get a local estimation due to the granularity. In other words, one document may have multiple approximated gains if they were returned at different ranks by the set of retrieval systems used. Take D_1 in Figure 6.3 as an example, it is ranked at the top position by S_1, S_4, S_5 and S_6, but not by S_2 or S_3. Suppose the relevance value of D_1 is not known, then it will have three different estimates when using a rank-level estimator. To summarize, rank-level estimators can approximate system performance to some extent by estimating an expected effectiveness gain of an unjudged document, albeit locally.

6.2.3 A Two-Stage Estimation Framework

A natural extension of rank-level estimators is to seek a solution to produce unified global estimations for unjudged documents.

Preliminaries. Consider the system matrix $\mathbf{S}_{k\times n}$ from the left hand side of Figure 6.3 where each column is a system and each row is a vector of documents. The main focus is to extend our method of local estimation presented in the previous section. To do this, we now consider a document-rank representation rather than the system-rank representation that was utilized for obtaining the rank-level estimators. Figure 6.3 indicates that, for each document, we can obtain its ranking information based on the set of contributing systems. This is also true of its own rank-level estimation vector. Let $\mathsf{k}_{i,j}$ be the rank of Document D_i given by system S_j and $g0_{j,m}$ be the rank-

k	S_1	S_2	S_3	S_4	S_5	S_6
1	D_1	D_2	D_7	D_1	D_1	D_1
2	D_2	D_{10}	D_{12}	D_2	D_3	D_2
3	D_8	D_5	D_8	D_4	D_5	D_4
4	D_{10}	D_7	D_3	D_5	D_8	D_7
...	...	...	...	...	...	...
10	D_7	D_1	D_6	D_7	D_{10}	D_{15}
...	...	...	...	...	...	...
100	D_{20}	D_3	D_{20}	D_{10}	D_{14}	D_{20}
...	...	...	...	...	...	...
1000	D_5	D_{40}	D_4	D_{20}	D_{500}	D_{55}

Get Doc-Rank Matrix →

	S_1	S_2	S_3	S_4	S_5	S_6
D_1	1	10	∞	1	1	1
D_2	2	1	∞	2	∞	2
D_3	∞	100	4	∞	2	∞
D_4	∞	∞	1000	3	∞	3
D_5	1000	3	∞	4	3	∞
D_6	∞	∞	10	∞	∞	∞
D_7	10	4	1	10	∞	4
D_8	3	∞	3	∞	4	∞
D_9	∞	∞	∞	∞	∞	∞
			...			

Figure 6.6: The ranking matrix for all documents (LHS), mapped from contributing systems (RHS) in Figure 6.3. The total number of rows is the number of documents in the collection and the total numbers of columns is the same as the number of contributing systems. An ∞ represents the document not appearing in top $1{,}000$ of the corresponding system.

level estimate using $G_m(j)$. For any document D, we can get its rank vector by $\mathsf{k}_{D,i}\,(1 \le i \le n)$, and corresponding rank-level estimations. For simplicity, we numbered the rank-level estimators and refer them using numbers henceforth. Consider D_1 in Figure 6.6 and suppose the third model is used, then immediately, we can represent D_1 as $\langle g0_{1,3}, g0_{10,3}, g0_{\infty,3}, g0_{1,3}, g0_{1,3}, g0_{1,3}\rangle$. We can consider all documents and possible estimators similarly. For simplicity, we use $g0_{*,m}$ to represent the rank-level gain vector for a document, when considering the m-th model G_m, and the subscript "$*$" represents $\langle \mathsf{k}_{D,1}, \mathsf{k}_{D,2}, \ldots, \mathsf{k}_{D,n}\rangle$ for a given document D (each row of the RHS matrix in Figure 6.6). We still focus on one topic and consider how to estimate system performance through estimating the background effectiveness gain of unjudged documents.

Overview of the Two-Stage Estimation Framework. To compute score estimates, we propose a two-stage framework, guided by a unified optimization goal, and built on a set of $m \ge 1$ per-topic rank-level estimators. The overall structure of this mechanism is described in Algorithm 5. We omit the process of obtaining rank-level estimations discussed in the previous section and assume as our starting point that m different rank-level estimators have been generated, each derived from the judged documents $D \in J'$, and that values for a set of gain functions have been computed, with $g0_{j,\ell}$ the gain associated with an unjudged document that appears in the j-th position of any of the n system rankings, as predicted by the ℓ-th of the m different estimators. Prior to forming the new combined estimates, we first compute the coefficient of covariance γ from the judgment set [22], in order to determine whether to use a background "unjudged are not relevant" predictor. Estimation is computed by steps 4 to 15, with $h_1(\cdot)$ and $h_2(\cdot)$ two parametric combining functions, in which the parameters are obtained by minimizing a loss function $L(\cdot)$. We discuss the details of Algorithm 5, including the rationale behind the use of γ, in the next few paragraphs.

Algorithm 5 Estimation Framework

Input: System matrix $\mathbf{S}_{k\times n}$; partial relevance judgments J' with $g2[D]$ the gain associated with document D for $D \in J'$ and undefined otherwise; and a set of m rank-level background gain estimates, $g0_{j,\ell}$ for $1 \le j \le k$ and $1 \le \ell \le m$, with $g0_{*,\ell} \equiv \langle g0_{j,\ell} \mid 1 \le j \le k\rangle$ and $g1_*[D] \equiv \langle g1_\ell[D] \mid 1 \le \ell \le m\rangle$.

Output: Values $g2[D]$, gain estimates for the documents $D \in J \setminus J'$

1: **for** $D \in J \setminus J'$ **do** $g2[D] \leftarrow 0$
2: $\gamma \leftarrow \text{COMPUTECV}(J', \mathbf{S})$ // compute coefficient of covariance
3: **if** $\gamma > \theta$ **then** // adjust only if γ exceeds threshold
4: **for** $\ell \leftarrow 1$ **to** m **do**
5: **for** $D \in J'$ **do** $g1_\ell[D] \leftarrow 0$
6: $\mathbf{w1}_{\text{opt}} \leftarrow \underset{\mathbf{w1}\in[0,1]^n}{\arg\min}\ L\left(h_1(g0_{*,\ell}, \mathbf{w1}) \mid D \in J'\right)$
7: **for** $D \in J$ **do**
8: $g1_\ell[D] \leftarrow h_1\left(g0_{*,\ell}, \mathbf{w1}_{\text{opt}}\right)$
9: **end for**
10: **end for**
11: $\mathbf{w2}_{\text{opt}} \leftarrow \underset{\mathbf{w2}\in[0,1]^m}{\arg\min}\ L\left(h_2(g1_*[D], \mathbf{w2}) \mid D \in J'\right)$
12: // get final per-document estimation
13: **for** $D \in J \setminus J'$ **do**
14: $g2[D] \leftarrow h_2\left(g1_*[D], \mathbf{w2}_{\text{opt}}\right)$
15: **end for**
16: **end if**
17: **return** $g2$

First Stage. As noted already, one problem with rank-level estimators is the potential inconsistency across runs of the gain attached to any particular document. In this stage, we only consider one rank-level estimator and aim at aggregating the n local estimations for each document to one estimation. That is, the m rank-level estimators are treated separately at first, in the loop at step 4, to obtain a consistent background gain for each document D for each model, denoted $g1_\ell[D]$. This is done via a combining function $h_1(\cdot)$ that maps a vector to a single value. There are plenty of design choices for implementing a combining function $h_1(\cdot)$, depending on different assumptions on relationships among n systems. For simplicity, we assume that the systems are independent and that they vary in quality. Therefore, for each document D, a natural combining function is to compute a weighted average, with h_1 (step 6) parameterized by an n-element weighting vector **w1** that is specific to the ℓ th estimator:

$$\begin{gathered}\forall D \in J',\ h_1(g0_{*,\ell}, \mathbf{w1} \mid D) = \sum_{i=1}^{n} g0_{\mathsf{k}_{D,i,\ell}} \cdot \mathbf{w1}_i \\ \text{with } \sum_{i=1}^{n} \mathbf{w1}_i = 1 \text{ and } \mathbf{w1}_i \in [0,1]\,,\end{gathered} \tag{6.6}$$

and where $g0_{\mathsf{k}_{D,i,\ell}}$ applies the ℓ-th estimator to the rank at which document D appears in the i-th of the n runs. One practical issue is that a document may not be retrieved by all systems in their top-k ranked lists, where k is the maximum depth of lists returned. In such cases the rank-based background gain of that document for that system is set to the modeled gain at depth k.

To compute a value for **w1**, we consider the aggregation process as an optimization problem, where the goal is to minimize the estimation error. We can view the estimation error from two granularities based on the two different representations in Figure 6.3 and Figure 6.6, respectively. On the set of judged documents, we can measure the difference between system effectiveness calculated using true and estimated relevance values, therefore, the first possible optimization goal is to minimize this difference. We refer to this loss as L_a. On the other hand, if we consider Figure 6.6, our goal can also be to approximate the true relevance value of a document, then the second loss function is the overall error for all documents in the training set. The loss described in the second case is modeled as L_b. We can use loss function L_a or L_b at step 6 of Algorithm 5 from either perspective.

In the first case, our aim is to minimize the estimation error of the rank-level estimators with regard to the true system performance; as such, the loss function L_a is defined as:

$$L_a(\cdot) = \sqrt{\sum_{i=1}^{n} \left(\sum_{\substack{j=1 \\ s_{j,i} \in J'}}^{k} (W_{\mathrm{M}}(j) \cdot (h_1(\cdot, \mathbf{w1} \mid s_{j,i}) - r_{j,i})) \right)^2}, \tag{6.7}$$

where J' is the set of judged documents and $h_1(\cdot, \mathbf{w1} \mid s_{j,i})$ is the combining function defined in Equation 6.6 with a rank-level estimator. Since all m rank-level estimators will be considered, we can assume the ℓ-th is considered in the current process iteration without loss of generality. Therefore, for system S_i and the pooling-to-d' judgment set J', the parameterized effectiveness score is

$$\sum_{\substack{j=1 \\ s_{j,i} \in J'}}^{k} W_{\mathrm{M}}(j) \cdot (h_1(\cdot, \mathbf{w1} \mid s_{j,i}) = \sum_{\substack{j=1 \\ s_{j,i} \in J'}}^{k} W_{\mathrm{M}}(j) \cdot (\sum_{i=1}^{n} g0_{\mathsf{k}_{s_{j,i},\ell}} \cdot \mathbf{w1}_i)$$

As noted, L_a minimizes the overall estimation error of the evaluation scores for the set of systems, which can be obtained directly by comparing the difference between the estimated and true system scores.

The alternative loss function L_b uses the document-position representation ($\mathsf{k}_{D,1}$,$\mathsf{k}_{D,2}$,..., $\mathsf{k}_{D,n}$):

$$L_b(\cdot) = \sum_{D \in J'} \sqrt{\sum_{i=1}^{n} (W_{\mathrm{M}}(\mathsf{k}_{D,i}) \cdot (h_1(\cdot, \mathbf{w1} \mid D) - r_D))^2}, \tag{6.8}$$

in which r_D is the relevance value of document D and is included only once for each document, rather than once per document-rank. This equation takes the representation in Figure 6.6 in to consideration and computes the difference between the estimated relevance and true relevance for each document in the judgment pool J'. When compared to Equation 6.7, which considers estimation errors at the system-level, this loss function is focused at the per-document level, seeking to minimize the overall estimation error for the weighted gain of each document. Either Equation 6.7 or Equation 6.8 is used at step 6 of Algorithm 5, with the combination function $h_1(\cdot)$ and constraints defined in Equation 6.6. The result is the computation of a sequence of $\mathbf{w1}_{\text{opt}}$ vectors, one for each of the m different rank-level estimators.

Second Stage. Multiple fitting models have been proposed because different assumptions about the underlying relevance distributions across all systems are plausible, with a risk that no single model covers the true hypothesis space. Indeed, the limited non-random training data means that we may suffer from a high variance if only one model is considered. In this case, it is unclear which model is better in the extrapolation stage. Since all proposed rank-level estimators capture the characteristics of a topic to some extent, a "meta" optimizer is used for combining results from the first stage, as described by steps 11 to 15. Similar to the first stage, we need to define a combining function in order to aggregate outputs from all models in the previous step.

A weighted average is used in this role too, considering each document D, together with the estimated background gains generated by the m previous computations, $g1_*[D]$. That combiner, $h_2(\cdot)$ (step 11), is defined via the m-vector **w2** as:

$$\begin{gathered}\forall D \in J',\ h_2\left(g1_*[D], \mathbf{w2}\right) = \sum_{\ell=1}^{m} g1_\ell[D] \cdot \mathbf{w2}_\ell\,, \\ \text{with } \sum_{\ell=1}^{m} \mathbf{w2}_\ell = 1 \text{ and } \mathbf{w2}_\ell \in [0,1]\,.\end{gathered} \tag{6.9}$$

Both L_a and L_b can be used in step 11, but may not necessarily be the same. Note that the m-vector $\mathbf{w2}_{\text{opt}}$, computed at step 11 as the minimizing value for Equation 6.9, provides an indication of the importance of individual optimizers from the previous stage. Previous work has shown that the expected error of combining loss functions is smaller than the average error on results output by each optimizer in isolation [133].

Computing the Coefficient of Covariance. Recall that our entire estimation framework is built on the assumption that *a shallow judgment pool cannot find a majority of relevant documents.* However, some topics may have only a small number of relevant documents, and a shallow depth may be sufficient to identify most of them, with adjustment unnecessary. When the shallow judgment pool is sufficient for identifying relevant documents, then assuming unjudged documents as not relevant is a good estimation for system performance. Only if deeper pooling would identify further relevant documents can score adjustment have an effect on system effectiveness scores. We propose the use of the *coefficient of covariance* [22] as an indicator, which is computed in step 2 of Algorithm 5.

We consider pooling as a sampling with replacement process, with an unknown probability of a relevant document being sampled. The judgment process removes duplicate documents from the original pool, ignoring the selection frequency of a document. The intuition behind γ is to make use of the frequency information to describe the sample coverage of relevant documents. In other words, if relevant documents have been selected with high frequency, then it is more probable that the pooling process is sufficient. Consider the system matrix $\mathbf{S}$ in Figure 6.3 and a pooling depth d'. Each document $s_{j,i}\,(1 \le j \le d',\ 1 \le i \le n)$ has a multiplicity in $\mathbf{S}_{d' \times n}$; we then group them by that frequency count. Let f_i be the number of relevant documents appearing i times in $\mathbf{S}_{d' \times n}$, $R' = \sum_i f_i$ the number of relevant documents, and $C = \sum_i i \cdot f_i$ be the total occurrence count of relevant documents. For example, if only D_8 and D_1 in Figure 6.3 are identified as

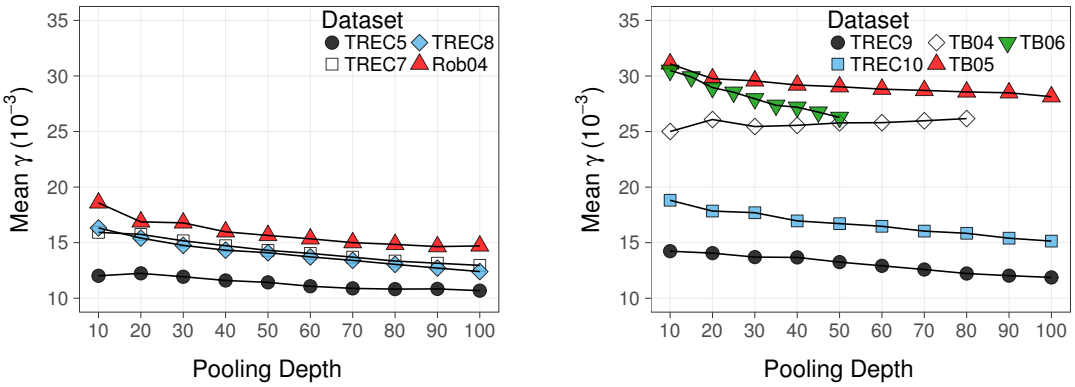


Figure 6.7: The average γ values on NewsWire (left) and Web (right) collections over topics $\times$ runs. The values are plotted against different pooling depth.

relevant documents, then we have $f_3 = 1$, $f_5 = 1$, and $R' = 2$ and $C = 8$. Based on these elements, the coefficient of covariance γ, is estimated via Chao and Lee [22]:

$$\gamma^2 = \max\left\{ \frac{\frac{|R'|}{1-f_1/C}\sum_i i\cdot(i-1)\cdot f_i}{C\cdot(C-1)} - 1, 0 \right\}. \tag{6.10}$$

When $\gamma = 0$, the probability of sampling a relevant document follows a uniform distribution; and when γ is high, the distribution is skewed, and it is likely that more relevant documents exist due to the low sampling coverage. Based on this, we have two hypotheses:

Hypothesis 1: *γ tends to decrease as pooling depth increases.*

Hypothesis 2: *There is a threshold θ, where if $\gamma < \theta$, then the existence of unjudged documents will only negligibly affect the estimate of the system performance, and they can be ignored.*

Average (over topics) γ values are plotted against pool depth in Figure 6.7, showing that γ decreases as the pooling depth increases. This is as expected, since the increasing pooling depth results in a more complete judgment set. Among the plotted datasets, TB04–TB06 have the largest average γ, which corresponds to a high relevance rate (Table 6.2). All earlier TREC collections have a smaller γ value since they are relatively "complete" compared to TB04–06. TREC-5 is a relatively complete test collection, and hence has the lowest γ among the datasets plotted.

6.3 Experiments with Rank-Level Estimators

6.3.1 Experimental Settings

We use RBP(0.95) throughout our experiments, a relatively deep metric (at an evaluation depth of 50, the inherent RBP(0.95) tail-residual is 0.07, and at an evaluation depth of 100, it is 0.006), and with the rank-level estimator Const $(G_c(\cdot))$ omitted since it behaves similar to LB. For each test collection, shallow judgment sets are constructed by pooling to depth d' ($d' < d$) using the set of identified contributing systems. Each run is then evaluated to depth $k = d$ (listed in Table 6.2)

using pooled-to-d' judgments, if they are available, or using estimated gain values $\hat{r}_j$ generated by the model for that topic. The RBP score estimate that results is then compared to the score and residual range generated using the full pool. If the extrapolated RBP score falls within that pooled-to-d range, an ϵ of zero is registered for that system-topic combination; if it falls outside the range, a non-zero ϵ is registered, as described in Section 6.1.3. Each value in the table is then the average over systems of the root-mean-square of that system's per-topic ϵ; with the parenthesized number beside it recording the percentage of the ϵ values that are zero, corresponding to predictions that fell within the final RBP score range. Besides RMSE of ϵ and Acc%, we also evaluate each method's ability for predicting system orderings using both Kendall's τ distance and the weighted distance defined in Equation 6.4. We compared our proposed methods to the following baselines:

- Lower Bound (LB) method: this estimation method assumes unjudged documents are not relevant thus indicates the lower bound performance that a system can achieve.
- Upper Bound (UB) method: in contrast to method LB, this method assumes all unjudged documents are relevant, giving the maximum performance score that a system can potentially achieve.
- The Ravana and Moffat [83] (RM) interpolative method defined in Equation 2.41.

We first explore the predictive power of rank-level estimators in Section 6.3.2, and then we consider whether using γ as an adjustment indicator helps in giving better per-topic adjustment results. We then show adjustment results using the two-stage adjustment framework described in Section 6.4.1 and make use of an adjustment threshold which is based on the rank-level estimation. In Section 6.4, we also demonstrate the applicability of the proposed two-stage or estimators for dealing with the system bias, and its ability for being applied with judgments constructed using a two-strata sampling approach.

6.3.2 Score Estimation using Rank-Level Estimators

All of the models are sensitive to the pooling depth d', and it is only when sufficient initial observations are available that it is appropriate to extrapolate. There is a clear difference depicted in Table 6.5 and 6.6, between the earlier TREC test collections, TREC-9, TREC-10 and Robust04 and the more recent TB04–06 test collections. The biggest distinction is that using the method LB is sufficient in the earlier test collections, as listed in Table 6.5. Only when the nominal pooling depth d' becomes larger, does the proposed rank-level estimators start to perform better than the LB method. This observation changes when we perform score adjustment on the TB04–06 test collections. On the TB04–06 test collections, shown in Table Table 6.6, LB performs worse than most of the rank-level estimators, resulting in large RMSE values and low Acc%. Also, there is no estimator that consistently outperforms the others and, all of the models are more sensitive to the initial judgment depth when compared using earlier TREC test collections. Also interesting in Table 6.5 is that the linear model, Lin ($G_\ell(\cdot)$), provides score predictions that are as reliable as those of the Weibull model. As a broad guidance, based on Table 6.5 and 6.6, we suggest that if an evaluation is to be carried out to depth k, then pooled judgments to depth $d' \geq k/2$ are desirable,

d'	TREC-9						
	LB	Static	Lin	Zipf	Wei	RM	UB
10	**0.031** (**45**)	0.157 (1)	0.037 (31)	0.048 (15)	0.041 (30)	0.057 (22)	0.329 (1)
20	**0.012** (**59**)	0.084 (2)	0.013 (51)	0.020 (32)	0.014 (42)	0.025 (32)	0.176 (1)
30	**0.006** (**67**)	0.046 (3)	0.006 (66)	0.009 (52)	0.006 (64)	0.012 (45)	0.010 (1)
40	**0.003** (75)	0.026 (9)	**0.003** (**77**)	0.004 (69)	0.003 (76)	0.006 (63)	0.056 (1)
50	**0.001** (82)	0.014 (22)	**0.001** (84)	0.002 (81)	**0.001** (**85**)	0.003 (76)	0.032 (2)
d'	TREC-10						
	LB	Static	Lin	Zipf	Wei	RM	UB
10	0.038 (**39**)	0.149 (1)	**0.034** (25)	0.051 (7)	0.038 (17)	0.064 (16)	0.318 (1)
20	0.016 (**53**)	0.079 (2)	**0.015** (45)	0.023 (25)	0.018 (37)	0.030 (25)	0.1680 (1)
30	**0.007** (**64**)	0.043 (4)	**0.007** (61)	0.011 (43)	0.009 (52)	0.015 (37)	0.093 (1)
40	**0.004** (**73**)	0.023 (9)	**0.004** (**74**)	0.005 (63)	0.004 (72)	0.007 (53)	0.052 (1)
50	**0.002** (78)	0.012 (21)	**0.002** (83)	**0.002** (81)	**0.002** (**84**)	0.0034 (70)	0.029 (2)
d'	Robust04						
	LB	Static	Lin	Zipf	Wei	RM	UB
10	0.046 (**21**)	0.1247 (1)	**0.043** (18)	0.069 (4)	0.049 (13)	0.088 (5)	0.277 (0)
20	0.020 (**34**)	0.066 (2)	**0.015** (33)	0.027 (15)	0.018 (32)	0.040 (9)	0.145 (0)
30	0.008 (49)	0.035 (3)	**0.007** (**52**)	0.012 (32)	0.008 (47)	0.020 (14)	0.078 (0)
40	0.004 (61)	0.019 (5)	**0.003** (**65**)	0.005 (56)	0.004 (**65**)	0.010 (31)	0.043 (0)
50	0.002 (69)	0.009 (15)	0.002 (75)	0.002 (71)	**0.001** (**78**)	0.004 (49)	0.024 (0)

Table 6.5: RMSE of ϵ and Acc% of rank-level Estimators using RBP(0.95). LB, RM [83] and UB are three baseline methods. $d\%$ are the nominal pooling depth and the reference depths are the original pooling depth listed in Table 6.2. Bold values are the best on that test collections at the specified pooling depth, and the numbers in parentheses are the percentage of the system-topic combinations for which $\epsilon = 0$ (point B in Figure 6.1). All evaluation depths are the same as the reference pooling depth on each collection.

d'	TB04						
	LB	Static	Lin	Zipf	Wei	RM	UB
10	0.117 (14)	0.144 (3)	**0.082** (**15**)	0.088 (12)	0.093 (13)	0.083 (14)	0.327 (1)
20	0.053 (21)	0.077 (9)	**0.039** (25)	0.043 (25)	0.041 (**27**)	0.039 (23)	0.180 (1)
30	0.026 (26)	0.041 (15)	**0.018** (39)	0.020 (39)	0.019 (**40**)	0.020 (39)	0.101 (3)
40	0.013 (34)	0.020 (27)	0.008 (56)	0.009 (58)	**0.008** (**60**)	0.009 (56)	0.056 (5)
50	0.007 (44)	0.009 (45)	0.004 (72)	**0.003** (**78**)	**0.003** (**78**)	0.004 (74)	0.030 (7)
d'	TB05						
	LB	Static	Lin	Zipf	Wei	RM	UB
10	0.125 (6)	0.108 (3)	0.085 (**7**)	0.086 (4)	0.082 (5)	**0.080** (5)	0.239 (1)
20	0.056 (10)	0.060 (6)	**0.039** (**13**)	0.045 (11)	0.041 (**13**)	0.041 (10)	0.132 (2)
30	0.028 (16)	0.034 (11)	**0.021** (**24**)	0.025 (21)	0.023 (**24**)	0.022 (18)	0.076 (3)
40	0.015 (23)	0.018 (18)	**0.011** (**41**)	0.013 (36)	0.011 (39)	0.012 (30)	0.043 (4)
50	0.007 (31)	0.010 (34)	0.005 (54)	0.006 (53)	**0.005** (**55**)	0.006 (42)	0.025 (6)
d'	TB06						
	LB	Static	Lin	Zipf	Wei	RM	UB
10	0.089 (24)	0.092 (25)	**0.059** (43)	0.068 (45)	0.061 (**46**)	0.065 (43)	0.249 (3)
15	0.051 (33)	0.059 (41)	**0.034** (57)	0.040 (**63**)	0.034 (60)	0.0373 (55)	0.177 (3)
20	0.033 (40)	0.036 (56)	0.021 (66)	0.022 (**75**)	**0.019** (72)	0.023 (68)	0.126 (5)
25	0.021 (49)	0.020 (68)	0.012 (75)	0.012 (84)	**0.010** (**83**)	0.014 (80)	0.087 (9)
30	0.0134 (58)	0.009 (80)	0.006 (85)	0.005 (**91**)	**0.005** (**91**)	0.007 (87)	0.059 (14)

Table 6.6: RMSE of ϵ and Acc% of rank-level Estimators using RBP(0.95), settings are the same as described in Table 6.5.

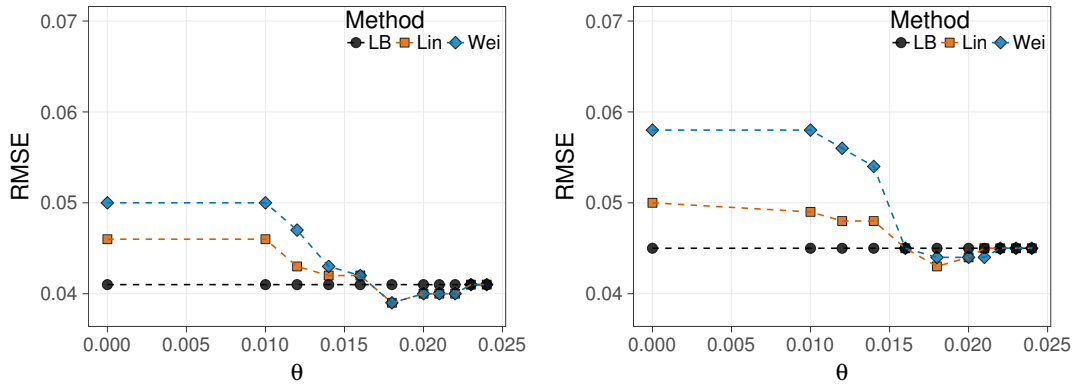


Figure 6.8: The relation between θ and RMSE of ϵ on TREC-5 and TREC-7, when $d' = 10$. A threshold value θ is the adjustment is set based on the γ statistic, and topics with $\gamma > \theta$. The RMSE of ϵ is the evaluation criterion same as previous. We show the impact on the two representative models: Linear (Lin) and Weibull (Wei) and on TREC-5 (left) and TREC-7 (right) test collections.

and that application of either the Weibull model Wei ($G_w(\cdot)$) or the simpler linear model Lin (G_ℓ) to infer any missing gain values between d' and k will lead to reliable final score outcomes. Both of these approaches outperformed the previous RM approach [83].

6.3.3 Determining the Adjustment Threshold

The idea of using γ as an adjustment indicator is introduced in Section 6.2.3, along with two hypotheses. The first hypothesis was discussed in Section 6.2.3 and plotted in Figure 6.7. In this section, we turn our attention to the second hypothesis.

To set the cutoff θ we use the earlier datasets TREC-5, TREC-7 and TREC-8 and perform a post-hoc analysis, noting that the majority of relevant documents have been identified in these collections [134], and hence that the computed RMSE should be close to the true error. Figure 6.8 shows the results for TREC-5 and TREC-7. Results on TREC-8 are omitted since they show a similar trend to the other two. We take $d' = 10$ as the training set for setting the adjustment threshold θ. The first hypothesis that γ becomes smaller when increasing the pooling depth, is confirmed in Figure 6.7. Results of the three models LB, Lin and Wei are shown in Figure 6.8. Weibull (Wei) may be an overestimate due to the shaping parameter, and Linear (Lin) tends to provide low estimates due to the monotonically decreasing nature of the model, as discussed in Section 6.2.2. At first, neither of the score estimation methods works better than the lower bound LB, but as θ increases, fewer topics need to be estimated, and when $\theta = 0.018$, both estimation methods outperform LB.

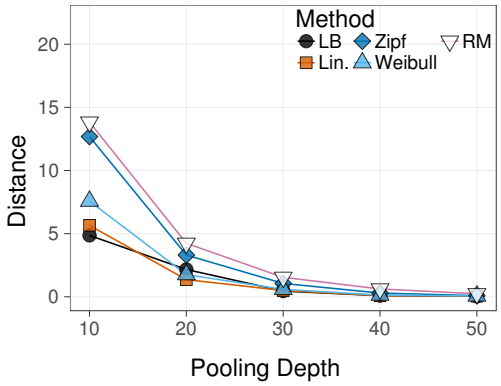
This observation confirms our second hypothesis that a small γ value may indicate the incompleteness of the judgments has small impact on the final effectiveness score, and it also suggests that score adjustment on a per-topic basis is applicable. We then apply this threshold $\theta = 0.018$ to other test collections when estimating system performance using rank-level estimators, as listed in Table 6.7.

$d\%$	TREC-9				TREC-10			
	LB	Lin	Zipf	Wei	LB	Lin	Zipf	Wei
10	**0.031** (**45**)	**0.031** (44)	0.033 (43)	**0.031** (44)	0.038 (**39**)	**0.033** (31)	0.042 (24)	0.036 (26)
20	**0.012** (59)	**0.012** (**60**)	0.013 (57)	**0.012** (59)	0.016 (53)	**0.014** (**51**)	0.017 (46)	0.015 (50)
30	**0.006** (67)	**0.006** (**68**)	0.006 (67)	**0.006** (**68**)	**0.007** (**64**)	**0.007** (63)	0.009 (57)	0.008 (60)
40	**0.003** (**75**)	**0.003** (**75**)	**0.003** (**75**)	**0.003** (**75**)	**0.004** (73)	**0.004** (**74**)	**0.004** (71)	**0.004** (**74**)
50	**0.001** (82)	**0.001** (**83**)	0.002 (82)	**0.001** (**83**)	**0.002** (78)	**0.002** (81)	**0.002** (81)	**0.002** (**83**)
$d\%$	Robust04				TB04			
	LB	Lin	Zipf	Wei	LB	Lin	Zipf	Wei
10	0.046 (**21**)	**0.040** (20)	0.048 (16)	0.043 (17)	0.117 (**14**)	**0.087** (**14**)	0.093 (12)	0.094 (13)
20	0.020 (**34**)	**0.016** (**34**)	0.019 (29)	0.017 (33)	0.053 (21)	**0.041** (25)	0.046 (24)	0.044 (**26**)
30	0.008 (49)	**0.007** (52)	0.008 (48)	**0.007** (**53**)	0.026 (26)	**0.019** (38)	0.021 (**39**)	**0.019** (**39**)
40	0.004 (61)	0.004 (63)	0.004 (63)	**0.003** (**65**)	0.013 (34)	**0.009** (54)	**0.009** (57)	**0.009** (**58**)
50	**0.002** (69)	**0.002** (72)	**0.002** (**74**)	**0.002** (**74**)	0.007 (44)	**0.004** (70)	**0.004** (**76**)	**0.004** (**76**)
$d\%$	TB05				TB06			
	LB	Lin	Zipf	Wei	LB	Lin	Zipf	Wei
10	0.125 (6)	0.085 (**7**)	0.086 (6)	**0.082** (**7**)	0.089 (24)	**0.059** (43)	0.068 (45)	0.061 (**46**)
20	0.056 (10)	**0.039** (13)	0.045 (12)	0.041 (**14**)	0.051 (33)	**0.034** (57)	0.040 (**63**)	0.034 (60)
30	0.028 (16)	**0.021** (**24**)	0.025 (21)	0.023 (**24**)	0.033 (40)	0.021 (66)	0.022 (**75**)	**0.019** (72)
40	0.015 (23)	**0.011** (**42**)	0.012 (38)	**0.011** (40)	0.021 (49)	0.012 (75)	0.012 (84)	**0.010** (**83**)
50	0.007 (31)	**0.005** (**54**)	0.006 (**54**)	**0.005** (**54**)	0.013 (58)	0.006 (85)	**0.005** (**91**)	**0.005** (**91**)

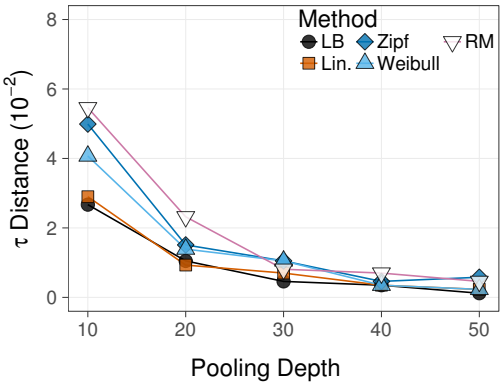
Table 6.7: Per-topic score estimation using rank-level estimators. Only topics with $\gamma > \theta$ are adjusted. Nominal pooling depth as a percentage of the original ones are listed in column $d\%$. All other settings are the same as Table 6.5

Compared to Table 6.5, applying the threshold improves the predicative power of rank-level estimators in general, especially for earlier TREC test collections such as TREC-9. For example, in Table 6.5, the LB outperforms all the other rank-level estimators in terms of RMSE on TREC-9. However, when we only perform the estimation on topics where $\gamma > 0.018$, both Lin and Wei become comparable to LB. On depths $d' \geq 10$. Lin outperforms LB when measuring Acc%. Similar observations can be made on Robust04 and TREC-10, where the RMSE of both Lin and Wei are reduced. Results on TB04–06 test collections were less affected. This is not surprising according to Figure 6.7, which shows higher γ values on the TB04–TB06 collections than earlier TREC datasets, meaning fewer relevant documents have been identified. The results in Table 6.7 also suggest that most of the topics on TB04–06 test collections should be adjusted, and the LB method is not a good estimator. The only unexpected observation occurs on the TB04 test collection, where the threshold falsely identifies Topic 734 as having a "sufficient" sampling of relevant documents, but around 48% in the final judged set are relevant, which increases the RMSE value Moreover, the results in both Table 6.5 and 6.7 further confirm that there is no rank-level estimator that consistently outperforms other estimators across all types of test collections using different pooling depths, though the method Lin and the method Wei may be a better approximation in most cases. That then leaves the choice of k, the evaluation depth to be used. As noted by Moffat and Zobel [74], k is in part determined by the properties of the user model that is embedded in the metric. In the RBP model used in all experiments, the persistence parameter $p = 0.95$ indicates a deep evaluation. When p is smaller and the user is considered to be less patient, the fact that the tail residual is given by p^k means that smaller values of k can be adopted to yield that same level of tail residual, as demonstrated in Chapter 5.

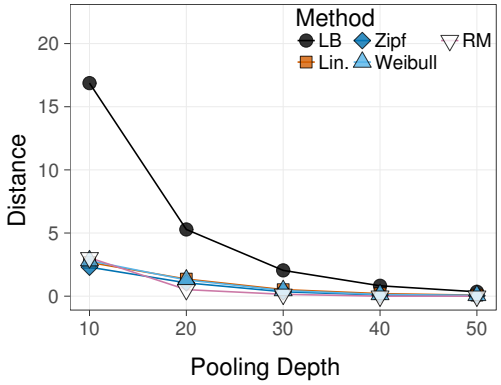
We measure both normalized τ distance and weighted distance (defined in Equation 6.4), all the outcomes are shown in Figure 6.9. Each row shows results from the same test collection, measured using the weighted distance (the left column) and the normalized τ distance (the right column). All distances are computed using the system orderings at the original pooling depth as the reference, and with the score estimation performed for all topics on a per topic basis. Results from TREC-9/10 are omitted here since they show a similar trend to Robust04. The RM approach performs well on TB04/05 but not on Robust04 , agreeing with the results in Table 6.5. This is because in the larger collections such as TB04/05/06, the gain decreases at a slower rate, making the assumptions in RM more appropriate. In contrast, LB approach provides more accurate predictions of system orderings on Robust04, but not on the TeraByte tasks. Among the rank-level estimators that were considered, the Lin method is more similar to the LB method; the Zipf method exhibits a similar behavior to the RM approach; and the Wei method shows a behavior between the two, being neither the best nor the worst on all test collections. However, as noted above, τ distance is sensitive to swaps that might be inconclusive. This can be observed by comparing the two columns in Figure 6.9, where the weighted distance is much smoother than the Kendall's τ distance.



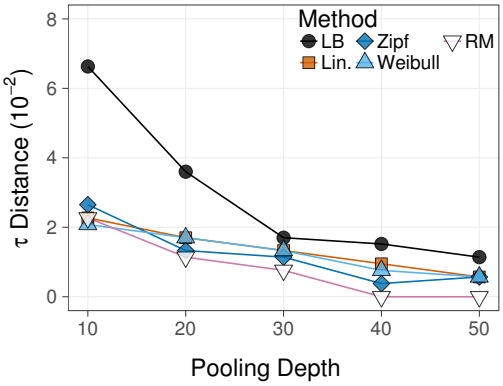
(a) Weighted distance on Rob04



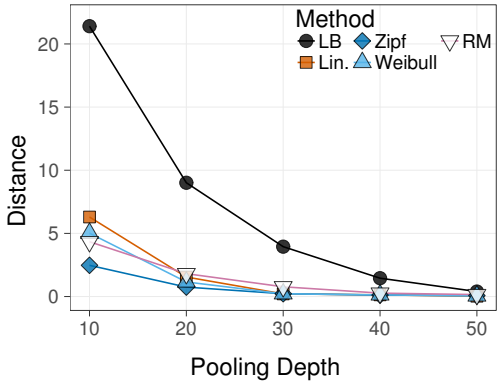
(b) Normalized τ distance on Rob04



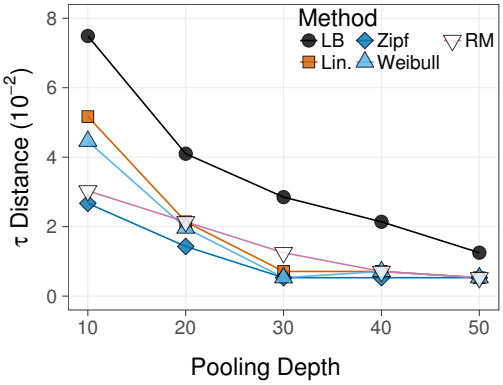
(c) Weighted distance on TB04



(d) Normalized τ distance on TB04



(e) Weighted distance on TB05



(f) Normalized τ distance on TB05

Figure 6.9: System orderings predicted using rank-level estimators on three test collections. The first column shows results measured using weighted distance defined in Equation 6.4 and the second column shows results measured using normalized τ distance. Results of Static, Const and UB are omitted since they are less competitive. All distance values are computed using the system ordering at original pooling depth as the reference.

d'	LB	RM	Lin		L_a		L_b	
			$\theta = 0$	$\theta = 0.018$	$\theta = 0$	$\theta = 0.018$	$\theta = 0$	$\theta = 0.018$
Robust04								
10	0.060 (19)	0.126 (4)	0.068 (11)	0.06 (16)	0.060 (9)	0.057 (16)	**0.050** (**19**)	**0.050** (18)
40	**0.011** (56)	0.052 (14)	0.039 (36)	0.017 (51)	0.015 (45)	**0.011** (53)	0.012 (**59**)	**0.011** (57)
100	0.004 (85)	0.027 (57)	0.024 (66)	0.007 (81)	0.004 (85)	0.003 (85)	0.004 (**87**)	**0.003** (86)
TB04								
10	0.181 (**11**)	0.202 (**11**)	0.131 (9)	0.140 (9)	**0.117** (**11**)	0.126 (**11**)	0.119 (**11**)	0.126 (**11**)
40	0.091 (23)	0.134 (25)	0.104 (23)	0.102 (23)	0.067 (28)	0.068 (30)	**0.064** (28)	0.066 (**31**)
80	0.063 (44)	0.104 (40)	0.090 (40)	0.084 (41)	0.050 (48)	0.049 (48)	**0.049** (**51**)	**0.049** (**51**)
TB05								
10	0.170 (6)	0.141 (5)	0.125 (4)	0.125 (5)	**0.110** (4)	**0.110** (6)	**0.110** (5)	**0.110** (**7**)
40	0.071 (19)	0.093 (18)	0.092 (22)	0.088 (23)	0.062 (26)	**0.061** (26)	0.063 (**28**)	0.063 (**28**)
100	0.035 (53)	0.066 (20)	0.073 (49)	0.070 (51)	0.035 (56)	0.035 (57)	**0.032** (**59**)	**0.032** (**59**)
TB06								
10	0.125 (22)	0.185 (32)	0.112 (35)	0.111 (35)	0.090 (**48**)	0.090 (47)	**0.086** (46)	0.087 (46)
20	0.078 (36)	0.163 (48)	0.104 (52)	0.100 (52)	0.068 (**65**)	0.068 (**65**)	**0.065** (62)	**0.065** (62)
50	0.041 (68)	0.121 (70)	0.100 (73)	0.095 (74)	0.042 (81)	0.042 (80)	**0.040** (**81**)	**0.040** (**81**)

Table 6.8: RMSE and Acc% for leave-group-out experiments with different depth d', averages across groups assuming that each group in turn is omitted from pool construction (RBP(0.95)).

6.4 Experiments with the Estimation Framework

6.4.1 Shallow Pooled Judgments

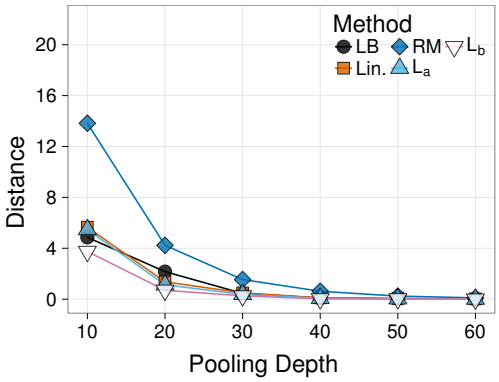
For this experiment we make use of the Lin method, the Zipf method and the Wei method as the rank-level estimators to derive a complete estimation framework, hence $m = 3$ is set in Algorithm 5. The two loss functions defined in Equations 6.7 and 6.8, denoted L_a and L_b respectively, are employed in the optimization framework. The same loss functions are used in both stages, and aggregated via Equations 6.6 and 6.9. All the other settings are same as described previously.

We show results for the Lin rank-level estimator again as a comparison in Table 6.9. In this experiment, we also investigate the impact of setting a threshold $\theta = 0.018$ on the score estimation, as discussed previously. The L_a approach has a higher RMSE than L_b and on earlier datasets (TREC-9, TREC-10 and Robust04) is slightly worse than the LB and Lin. baselines. That is, the loss function L_a provides poorer coverage of the true hypothesis space than does L_b. However, for TB06, smaller RMSE (and larger Acc%) values are achieved when compared to the other collections. This is because the reference depth $d = 50$ is smaller, and the residuals are larger.

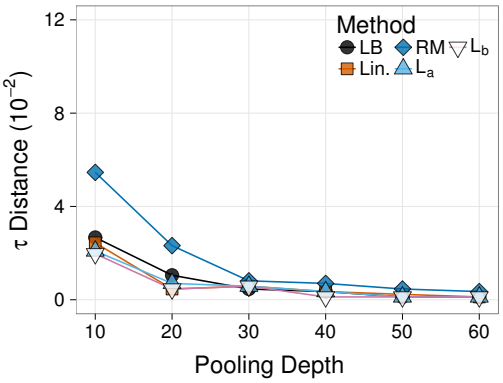
Similar to the previous experiments, we also compute the distance using both weighted distance and normalized Kendall's τ distance. The left column of Figure 6.10 shows the weighted

Dataset	d'	Lin.		L_a		L_b	
		$\theta = 0$	$\theta = 0.018$	$\theta = 0$	$\theta = 0.018$	$\theta = 0$	$\theta = 0.018$
TREC-9							
	10	0.037 (31)	0.031 (44)	0.038 (26)	**0.030** (44)	0.031 (41)	0.031 (**46**)
	20	0.013 (51)	0.012 (60)	0.013 (42)	0.012 (58)	**0.010** (**62**)	0.012 (61)
	30	0.006 (66)	0.006 (68)	**0.005** (63)	0.006 (68)	**0.005** (**71**)	0.006 (68)
	40	0.003 (77)	0.003 (75)	**0.002** (**79**)	0.003 (76)	**0.002** (**79**)	0.003 (76)
	50	**0.001** (84)	**0.001** (83)	**0.001** (**88**)	**0.001** (85)	**0.001** (86)	**0.001** (84)
TREC-10							
	10	0.034 (25)	0.033 (31)	0.036 (13)	0.031 (27)	**0.027** (34)	0.028 (38)
	20	0.015 (45)	0.014 (51)	0.016 (36)	0.014 (50)	**0.012** (53)	**0.012** (**55**)
	30	0.007 (61)	0.007 (63)	0.007 (55)	0.007 (62)	**0.006** (**66**)	**0.006** (**66**)
	40	0.004 (74)	0.004 (74)	**0.003** (75)	**0.003** (76)	**0.003** (**78**)	**0.003** (77)
	50	0.002 (83)	0.002 (81)	**0.001** (**86**)	0.002 (83)	**0.001** (84)	0.002 (82)
Robust04							
	10	0.043 (**21**)	0.039 (20)	0.045 (9)	0.039 (17)	0.039 (20)	**0.035** (20)
	20	0.015 (33)	0.016 (34)	0.016 (26)	0.015 (32)	**0.013** (**38**)	0.016 (35)
	30	0.007 (49)	0.007 (52)	0.007 (47)	0.006 (53)	**0.005** (**59**)	0.006 (55)
	40	0.003 (60)	0.004 (63)	**0.002** (68)	0.003 (67)	**0.002** (**73**)	0.003 (66)
	50	0.002 (69)	0.002 (72)	**0.001** (**84**)	**0.001** (76)	**0.001** (79)	0.002 (73)
TB04							
	10	0.082 (15)	0.087 (14)	**0.072** (15)	0.077 (15)	0.073 (**16**)	0.077 (**16**)
	20	0.039 (25)	0.041 (25)	0.035 (28)	0.037 (28)	**0.033** (**32**)	0.036 (31)
	30	0.018 (39)	0.019 (38)	**0.015** (**45**)	0.016 (44)	**0.015** (44)	0.016 (43)
	40	0.008 (56)	0.009 (54)	**0.007** (**65**)	**0.007** (62)	**0.007** (59)	0.008 (57)
	50	0.004 (72)	0.004 (70)	**0.003** (**80**)	**0.003** (78)	**0.003** (74)	0.004 (72)
TB05							
	10	0.085 (7)	0.085 (7)	0.070 (6)	0.070 (8)	**0.067** (7)	**0.067** (**9**)
	20	0.039 (13)	0.039 (13)	0.034 (14)	0.034 (14)	**0.033** (18)	**0.033** (**19**)
	30	0.021 (24)	0.021 (24)	0.018 (24)	0.018 (24)	**0.017** (**29**)	**0.017** (**29**)
	40	0.010 (41)	0.010 (41)	**0.008** (42)	**0.008** (41)	0.009 (**44**)	0.009 (**44**)
	50	0.005 (54)	0.005 (54)	**0.004** (**61**)	**0.004** (60)	**0.004** (59)	**0.004** (58)
TB06							
	10	0.059 (43)	0.059 (43)	**0.047** (**55**)	**0.047** (**55**)	0.053 (51)	0.053 (50)
	15	0.034 (57)	0.034 (57)	**0.026** (**70**)	**0.026** (**70**)	0.028 (62)	0.028 (62)
	20	0.021 (66)	0.021 (66)	**0.013** (**81**)	**0.013** (**81**)	0.017 (73)	0.017 (73)
	25	0.012 (75)	0.012 (75)	**0.007** (**89**)	**0.007** (**89**)	0.010 (81)	0.010 (81)
	30	0.006 (85)	0.006 (85)	**0.003** (**94**)	**0.003** (**94**)	0.005 (89)	0.005 (89)

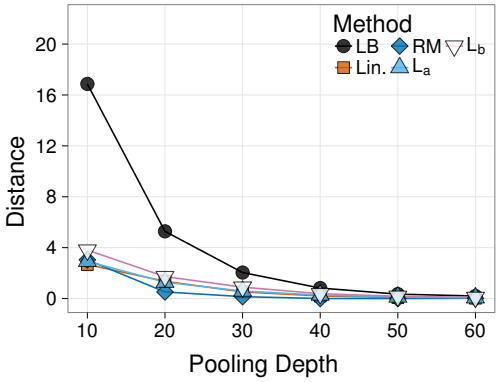
Table 6.9: RMSE of ϵ and Acc% using two-stage estimation framework. The evaluation metric is RBP(0.95) and the reference depth for each collection is the original pooling depth listed in Table 6.2. Bold numbers are the lowest RMSE and highest Acc% for that collection at that depth.



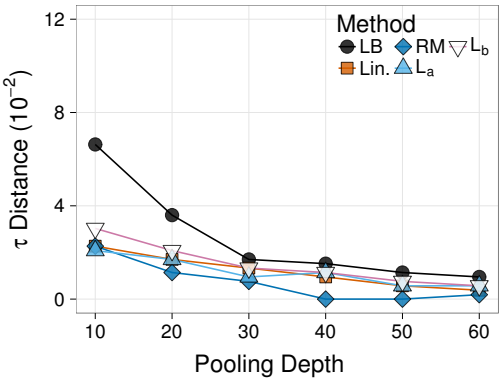
(a) Weighted distance on Rob04



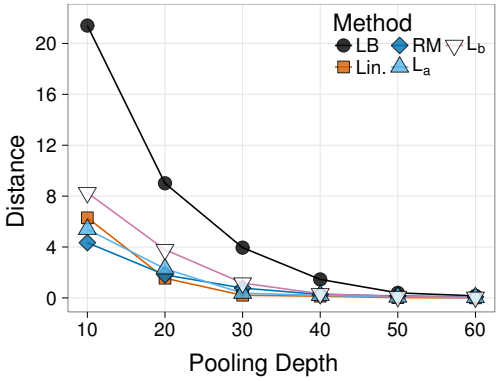
(b) Normalized τ distance on Rob04



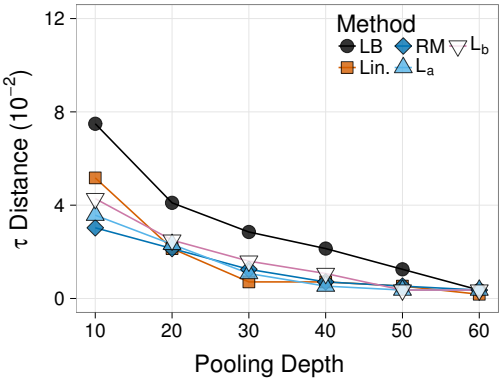
(c) Weighted distance on TB04



(d) Normalized τ distance on TB04



(e) Weighted distance on TB05



(f) Normalized τ distance on TB05

Figure 6.10: System orderings predicted using two-stage estimators on three test collections. All settings are the same as Figure 6.9.

distance measure of Equation 6.4. In the second column, when normalized Kendall's τ distance is measured, the estimation framework gives orderings close to the reference ordering across a range of nominal pool depths d'. Overall, there are situations in which LB performs poorly, and situations in which RM performs poorly. The Lin., L_a, and L_b methods consistently provide the highest agreements.

Although our estimation method is motivated by the issues associated with pooling depth bias, we also examined its applicability with regards to system bias. Table 6.8 lists results on the leave-group-out experiments, when the pooling depths vary $d' \in \{10, 40\}$, on Robust04, TB04 and TB05; and $d' \in \{10, 20\}$ on TB06 test collection, also, an original pool is also considered in the experiment. We can see that the proposed methods L_a and L_b still perform well compared to three baselines and on all considered datasets. However, when $d' = 10$ and for the TeraByte datasets, all methods exhibit a similar estimation error, and none is significantly better than all the others. Also we can observe that the RM is not a good choice when adjusting system bias compared to the alternatives. Moreover, although the proposed methods are applicable in adjusting system bias, their estimation errors are higher than when applying it to solve the pooling depth bias shown in Table 6.9. How can we further improve the estimation accuracy when system bias exists remains an open problem and a direction for future work.

6.4.2 Sampling-Based Judgments

In addition to the pooling-to-d' setting, we also examined the estimation process on the set of judgments selected using a two-strata sampling approach [117]. The two-strata sampling is arranged as a initial $d' = 10$ shallow pool, plus a 10% sampling of remaining documents to depth 100. The sampling process was repeated ten times, and the average statistics were already reported in Table 6.2. On this set of judgments we compute InfAP using trec_eval, and the InfRBP approach defined in Equation 6.1 derived from the RM method.

Figure 6.11 shows that L_a, L_b and InfRBP give rise to stable system orderings, with normalized τ distance scores below 0.05 across all collections. When the weighted distance is measured, L_a outperforms InfRBP on all collections but TREC-9, while L_b outperforms InfRBP except on TB05. The slightly worse outcome for L_a on TREC-9 is a consequence of the increase in the number of significantly different system pairs. Note the more variable outcomes generated when InfAP is used as the metric driving the system orderings.

6.4.3 Estimating Relevance on ClueWeb

To further explore the merit of our approach, we examine the CW09A-2010 collection. It has a shallow pool depth ($d = 20$), meaning that validation is not possible, as there is no deep-pool reference ordering. Instead, we compute the normalized τ distance between each pair of estimation methods, and simply record how much the rankings differ, shown in Figure 6.12. The UB estimator assumes that all unjudged documents are relevant. As a reference point, we also compute the same values for TB05, at two depths, $d' = 20$ and $d' = 60$. At the latter depth all estimation approaches tend to agree with each other. On TB05, all of the estimation results,

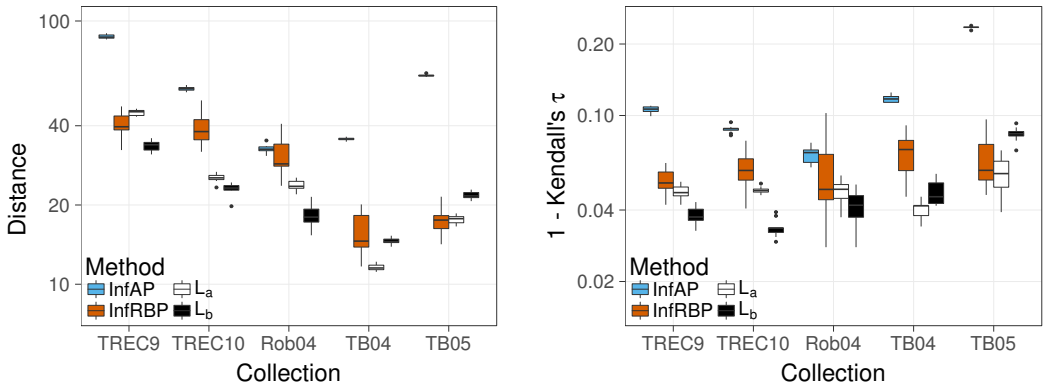


Figure 6.11: System ordering comparisons on a two-strata sampled judgment set, repeated ten times. Judgments are to depth $d' = 10$, plus a 10% random sample of remaining documents to depth 100 to form the second stratum. Note the logarithmic vertical scales.

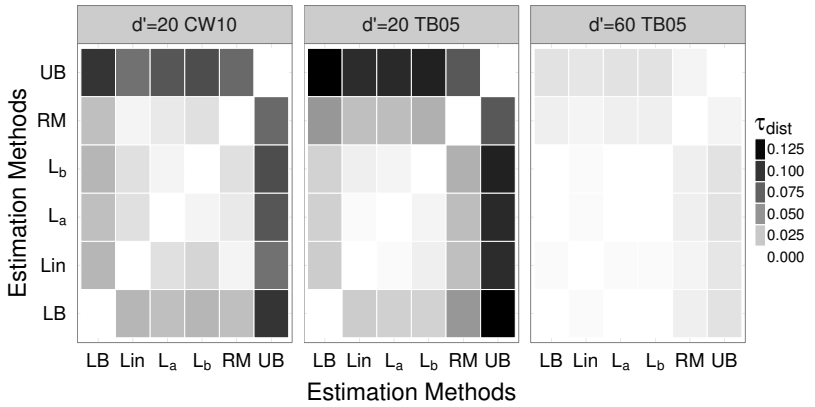


Figure 6.12: Normalized τ distance between system orderings generated by different estimation methods using a pool of depth $d' = 20$ on CW09A-2010 (CW10), and on TB05 with pool depths of $d' = 20$ and $d' = 60$.

including UB, tend to agree on the system ordering. However, on CW09A-2010, there is a clear uncertainty, confirming that $d' = 60$ is a more robust pool depth for TB05 than is $d' = 20$ on either TB05 or CW09A-2010 when seeking to apply RBP(0.95) as an evaluation metric. Great caution should be exercised when the $d = 20$ judgments are used for anything other than shallow metrics.

6.5 Conclusions

We have presented a new estimation framework to improve system comparisons in batch IR evaluation, with the key idea being to predict a gain value for each unjudged document. The entire process consists two components: a set of rank-level estimators and a two-stage estimator formulated as an optimization problem. The two-stage estimation is based on the rank-level estimators and it provides a unified global gain for each unjudged document, whilst rank-level estimators

only consider a local gain for each document, ignoring the fact that a document may be associated with multiple different retrieval ranks.

We show that estimation is a viable technique to predict scores for deep evaluation metrics when limited judgments are available, including the case when the judgments are obtained using stratified sampling rather than pooling. One important aspect of our approach is to make decisions on *when* to adjust topics, instead of treating all topics equally. By making use of the coefficient of covariance (γ) [22], we verified our proposed hypothesis – that whether a topic requires score estimation can be predicted via considering the sample coverage, measured using γ. We show that adjusting a subset of topics on earlier TREC test collections can improve the estimation accuracy; and it confirms the earlier statement that on these test collections, a majority of relevant documents can be identified.

A secondary contribution is the development of a new technique to more precisely compare system orderings. By focusing on swaps that are conclusive, our weighted rank correlation coefficient *dist* can be used to measure the stability of a variety of estimation techniques. Our experimental results show that, solely focusing on the significant swaps can give a more robust comparison of system orderings and that treating all swaps equally may make the results too sensitive to be conclusive. Using *dist*, we show that estimation improves our ability to score and compare systems using limited judgments.

It must be noted, however, that the estimation is built on the m rank-level fitted models, each of which requires that when constructing the judgment set, documents up to some rank d' be fully judged. This means that for some sampling-based judgment approaches, the proposed method is not applicable. Second, while we show that our estimation methods can also account for system bias to some extent, outcomes might be further improved by introducing more randomization into the optimization framework. Overall, there remain many evaluation challenges in this area, despite the gains that have been achieved here.

CHAPTER 7

Conclusions and Future Work

The focus of IR studies is to develop efficient and effective retrieval systems. The strategy of efficient query evaluation, the model for effective retrieval and the approaches for evaluating current systems are all important, and highly related to each other. Many questions are considered in the process of delivering a better retrieval system. Whether or not a retrieval system can provide a high quality ranked list to the user, for which a key factor is the heuristics or features employed within the model. In the long history of developing IR systems, many features have been examined, among these for which the proximity (or term-dependency) was shown to be important.

However, this general observation makes sophisticated models usually require more computing effort, and so for the term-dependency models to make use of higher-order proximity features. How can we efficiently to compute higher-order proximity statistics? What are the possible efficiency and effectiveness trade-offs when using higher-order term-dependency models? As mentioned, to examine efficiency and effectiveness, large-scale evaluation is required to be carried out, although with incomplete judgments. Therefore, several questions arise in this process. How reliable are the evaluation results on large-scale web test collections with limited judgment effort? How can we draw reliable conclusions when using current evaluation configurations? This thesis has contributed answers to these questions: the eager and lazy versions of PLANESWEEP for finding all higher-order proximity features in one-pass (Chapter 3); the use of distance-based local statistics as a trade-off between effectiveness and efficiency when considering higher-order proximity feature (Chapter 4); that weighted-precision metrics considering only top ranked documents such as ERR or RBP(0.8) should be employed on large test collections with a shallow pooling depth (Chapter 5); and a score estimation framework to adjust evaluation results for deep effectiveness metrics on shallow pooled test collections (Chapter 6).

In the process of developing new techniques for better satisfying users' information needs, there remain many open problems. We must continually seek answers to new questions that arise from the entire development life-cycle of IR systems that encompass algorithm design to evaluation and testing feedback for furthering the field. With the vast amounts of data these systems must today ingest, process and organize, we are faced with the issue of rediscovering new solutions to old problems.

7.1 Thesis Outcomes

The thesis began by proposing two PLANESWEEP variants for the efficient computation of higher-order term-dependency statistics of all subqueries (Chapter 3). Among all possible retrieval heuristics, proximity features, which assume terms have a certain level of dependency between terms, are known to be effective. It is also known that when considering all term-dependencies of subqueries having a length greater than or equal to two, the computational cost is high. Therefore, we empirically studied the effectiveness of higher-order proximity features in MRF-based term-dependency models [73], especially considering the difference between phrases and unordered windows. As our results showed, using either type of proximity statistics may help in improving retrieval effectiveness, which further motivated us for finding efficient solutions of computing these features. Sadakane and Imai [92] propose the PLANESWEEP algorithm that computes proximity statistics for each subquery consisting of more than two query terms. However, the exponential growth in the amount of subqueries leads to increased overhead, since the extraction process is repeated across all subqueries. Our proposed method overcomes this weakness and performs the extraction by evaluating the original query only once for each document.

A second bottleneck when computing higher-order term-dependency models is the requirement for global statistics, implying a two-pass evaluation on the index structure. We show and demonstrate in Chapter 4 that a variant of an existing MRF-based term-dependency model can be adopted, and local proximity statistics can be used as a surrogate of global proximity statistics. By doing so, we show that although the effectiveness may somewhat be degraded, avoiding unnecessary second evaluation on the index helps to reduce the computational cost.

We empirically examined the effectiveness of higher-order proximity models, and demonstrate their applicability on large test collections. Nevertheless, the incompleteness of these large test collections raise concerns of the reliability of the evaluation results. Shallow pooling depths lead to a high level of uncertainty in system comparison, when using deep metrics to evaluate systems. Different evaluation configurations are possible, and may vary in their conclusion of system performance. In Chapter 5, we experimented with various test collections, ranging in their different sizes and number of judged documents. We showed that the configuration that consists of pooling depth, evaluation depth and metric affect the final conclusions drawn from the evaluation process. More specifically, if only a shallow pooling depth is available on large test collections such as web-based collections, there are potential risks arising from the failure of identifying enough relevant documents. Missing a relatively large fraction of the relevant documents may cause an imprecise system performance evaluation, especially if recall-based or deep metrics are employed. We concluded that setting a proper evaluation depth and using the right metrics with the consideration of test collection is important. More specifically, on these large test collections with limited pooling depth, a shallow weighted-precision metric with residual information is recommended.

If there are situations where deep metrics must be used, estimation of the effectiveness score is required. We considered a possible solution to this problem in Chapter 6 and proposed an estimation framework for this problem. The estimation framework contains two components: one

that models effectiveness scores as a function of retrieval ranks and another that gives the estimates using a two-step optimization. We concluded that, it is possible to estimate system performance using our proposed framework. However, the question of how we can reduce evaluation bias when only a limited amount of judgments are available remains an open research direction.

7.2 Future Work

7.2.1 Extension of Efficient Higher-Order Proximity-Based Retrieval

In this thesis, we specifically consider proximity features used in retrieval models, from both effectiveness and efficiency perspectives. As our study shows, query-time features are expensive, while they may improve retrieval effectiveness to some extent. Two further questions then arise. From an effectiveness standpoint, the proximity features cause some queries to fail and there are numerous reasons as to why this is the case. Therefore, the first possible extension is to consider a dynamic mechanism where we can choose which queries to apply proximity-based models to. A second direction is related to both effectiveness and efficiency. So far, we considered the proximity features of all subqueries, aiming at either modeling a full dependency among queries or modeling a sequential dependency that indeed considers only bigrams in the query. This is neither effective nor efficient, in the sense that not all subqueries should be considered as valid dependencies among query terms. Although Weighted SDM (WSDM) is proposed and shown to be more effective [9], it would also be interesting if we can generalize it to a full dependency model.

When considering the efficiency issues in retrieval associated with proximity-based models, we observe that the computational overhead is also an ongoing issue that requires some efficient query evaluation strategies. Dynamic pruning techniques such as Weak AND (WAND) [12] may be considered as a starting point, and the upperbound estimation is the key of developing a new traversal algorithm. An alternative approach to further improve the retrieval efficiency is to exploit a multi-stage retrieval strategy [120] and assess trade-offs at different ranking stages.

7.2.2 Challenges in Reliable Evaluation on Large Test Collections

The observations and experimental results in Chapter 5 and Chapter 6 indicated the risk of drawing reliable conclusions when only a limited number of documents have been judged. Moreover, we saw that in Chapter 5, evaluation configurations are crucial when different test collections are considered. However, in some cases when extended evaluation is required or evaluating new retrieval systems, the incompleteness of judgments becomes a critical problem. Although our proposed method in Chapter 6 tackled the problem of reliable evaluation and approximated true system performance to some extent, the problem is far from being solved. As only weighted-precision metrics were considered in our study, the obvious question is, how can we estimate metric scores for recall-based effectiveness measurement? Although various studies consider the problem [51, 127, 130], their approaches require a new test collection building regime, hence result in a new question to be considered – *how can we build a cost-effective reusable test collection*? Even if we limited the metric to be weighted-precision, the estimation quality of our approach on

new retrieval systems still has space for considerable improvement. One possible way of further improving our existing work is to add randomization and apply drop-out techniques to reduce variance, which may help when estimating new systems that do not contribute to the pooling process.

Many interesting open questions remain when attempting to build efficient and effective retrieval systems.

Bibliography

[1] G. Amati. *Probability Models for Information Retrieval Based on Divergence from Randomness*. PhD thesis, University of Glasgow, 2003.

[2] V. N. Anh and A. Moffat. Pruned query evaluation using pre-computed impacts. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 372–379, 2006.

[3] N. Asadi and J. Lin. Document vector representations for feature extraction in multi-stage document ranking. *Information Retrieval J.*, 16(6):747–768, 2013.

[4] J. A. Aslam, V. Pavlu, and E. Yilmaz. A sampling technique for efficiently estimating measures of query retrieval performance using incomplete judgments. In *Proc. ICML Wrkshop. on Learning with Partially Classified Training Data*, pages 57–66, 2005.

[5] J. A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 541–548, 2006.

[6] P. Bailey, A. Moffat, F. Scholer, and P. Thomas. User variability and IR system evaluation. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 625–634, 2015.

[7] J. D. Beer and M. Moens. Rpref: A Generalization of Bpref towards graded relevance judgments. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 637–638, 2006.

[8] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 491–498, 2008.

[9] M. Bendersky, D. Metzler, and W. B. Croft. Learning concept importance using a weighted dependence model. In *Proc. Int. Conf. on Web Search and Data Mining*, pages 31–40, 2010.

[10] M. Bendersky, D. Metzler, and W. B. Croft. Parameterized concept weighting in verbose queries. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 605–614, 2011.

[11] A. Z. Broder. A taxonomy of web search. *ACM SIGIR Forum*, 36(2):3–10, 2002.

[12] A. Z. Broder, D. Carmel, H. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In *Proc. Int. Conf. Information and Knowledge Management*, pages 426–434, 2003.

[13] A. Broschart and R. Schenkel. High-performance processing of text queries with tunable pruned term and term pair indexes. *ACM Trans. Information Systems*, 30(1):1–32, 2012.

[14] C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 33–40, 2000.

[15] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 25–32, 2004.

[16] C. Buckley, D. Dimmick, I. Soboroff, and E. M. Voorhees. Bias and the limits of pooling for large collections. *Information Retrieval J.*, 10(6):491–508, 2007.

[17] S. Büttcher, C. L. A. Clarke, and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 621–622, 2006.

[18] S. Büttcher, C. L. A. Clarke, P. C. K. Yeung, and I. Soboroff. Reliable information retrieval evaluation with incomplete and biased judgements. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 63–70, 2007.

[19] B. Carterette and J. Allan. Semiautomatic evaluation of retrieval systems using document similarities. In *Proc. Int. Conf. Information and Knowledge Management*, pages 873–876, 2007.

[20] B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 268–275, 2006.

[21] B. Carterette, E. Kanoulas, V. Pavlu, and H. Fang. Reusable test collections through experimental design. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 547–554, 2010.

[22] A. Chao and S. Lee. Estimating the number of classes via sample coverage. *J. of the Am. Statistical Association*, 87(417):210–217, 1992.

[23] O. Chapelle, D. Metzler, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proc. Int. Conf. Information and Knowledge Management*, pages 621–630, 2009.

[24] C. L. A. Clarke, G. V. Cormack, and E. A. Tudhope. Relevance ranking for one to three term queries. *Information Processing & Management*, 36(2):291–311, 2000.

[25] C. L. A. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2004 Terabyte Track. In *Proc. Text REtrieval Conf.*, pages 74–82, 2004.

[26] C. L. A. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2009 Web Track. In *Proc. Text REtrieval Conf.*, pages 1–9, 2009.

[27] C. Cleverdon. The Cranfield tests on index language devices. *Aslib proceedings*, 19(6): 173–194, 1967.

[28] G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. Efficient construction of large test collections. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 282–289, 1998.

[29] G. V. Cormack, O. Lhotak, and C. R. Palmer. Estimating precision by random sampling. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 273–274, 1999.

[30] G. V. Cormack, M. D. Smucker, and C. L. A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval J.*, 14(5):441–465, 2011.

[31] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *J. of Algorithms*, 55(1):58–75, 2005.

[32] W. B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison Wesley, Boston, 2010.

[33] J. S. Culpepper, M. Petri, and F. Scholer. Efficient in-memory top-k document retrieval. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 225–234, 2012.

[34] O. de Kretser and A. Moffat. Effective document presentation with a locality-based similarity heuristic. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 113–120, 1999.

[35] T. Elsayed, J. Lin, and D. Metzler. When close enough is good enough: Approximate positional indexes for efficient ranked retrieval. In *Proc. Int. Conf. Information and Knowledge Management*, pages 1993–1996, 2011.

[36] J. L. Fagan. *Experiments in automatic phrase indexing for document retrieval: a comparison of syntactic and non-syntactic methods*. PhD thesis, Cornell University, 1987.

[37] H. Fang, T. Tao, and C. Zhai. A formal study of information retrieval heuristics. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 49–56, 2004.

[38] L. Grönqvist. Evaluating latent semantic vector models with synonym tests and document retrieval. In *Proc. ELECTRA Wrkshop.*, pages 86–88, 2005.

[39] D. K. Harman. Overview of the second text retrieval conference (TREC-2). In *Proc. Text REtrieval Conf.*, pages 351–357, 1994.

[40] D. Hawking. Overview of the TREC-9 web track. In *Proc. Text REtrieval Conf.*, pages 87–102, 1999.

[41] D. Hawking and S. E. Robertson. On collection size and retrieval effectiveness. *Information Retrieval J.*, 6(1):99–105, 2003.

[42] D. Hawking and P. Thistlewaite. Proximity operators: so near and yet so far. In *Proc. Text REtrieval Conf.*, pages 131–143, 1995.

[43] B. He, J. X. Huang, and X. Zhou. Modeling term proximity for probabilistic information retrieval models. *Information Sciences*, 181(14):3017–3031, 2011.

[44] D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 329–338, 1993.

[45] S. Huston and W. B. Croft. A comparison of retrieval models using term dependencies. In *Proc. Int. Conf. Information and Knowledge Management*, pages 111–120, 2014.

[46] S. Huston, J. S. Culpepper, and W. B. Croft. Sketch-based indexing of n-words. In *Proc. Int. Conf. Information and Knowledge Management*, pages 1864–1868, 2012.

[47] S. Huston, J. S. Culpepper, and W. B. Croft. Indexing word-sequences for ranked retrieval. *ACM Trans. Information Systems*, 32(1):3, 2014.

[48] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 41–48, 2000.

[49] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Information Systems*, 20(4):422–446, 2002.

[50] G. K. Jayasinghe, W. Webber, M. Sanderson, and J. S. Culpepper. Extending test collection pools without manual runs. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 915–918, 2014.

[51] E. Kanoulas and J. A. Aslam. Empirical justification of the gain and discount function for NDCG. In *Proc. Int. Conf. Information and Knowledge Management*, pages 611–620, 2009.

[52] J. Kekäläinen and K. Järvelin. Using graded relevance assessments in IR evaluation. *J. of the Soc. for Information Science and Technology*, 53(13):1120–1129, 2002.

[53] R. Kumar and S. Vassilvitskii. Generalized distances between rankings. In *Proc. Int. Conf. World Wide Web*, pages 571–580, 2010.

[54] L. Lee. IDF revisited: A simple new derivation within the Robertson-Spärck jones probabilistic model. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 751–752, 2007.

[55] A. Lipani, M. Lupu, and A. Hanbury. Splitting water: Precision and anti-precision to reduce pool bias. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 103–112, 2015.

[56] A. Lipani, M. Lupu, and A. Hanbury. The curious incidence of bias corrections in the pool. In *Proc. European Conf. on Information Retrieval*, pages 267–279, 2016.

[57] A. Lipani, M. Lupu, E. Kanoulas, and A. Hanbury. The solitude of relevant documents in the pool. In *Proc. Int. Conf. Information and Knowledge Management*, pages 1989–1992, 2016.

[58] A. Lipani, G. Zuccon, M. Lupu, and B. Koopman. The impact of fixed-cost pooling strategies on test collection bias. In *Proc. Int. Conf. Theory of Information Retrieval*, pages 105–108, 2016.

[59] X. Lu, A. Moffat, and J. S. Culpepper. How effective are proximity scores in term dependency models? In *Proc. Australasian Doc. Comp. Symp.*, pages 89–90, 2014.

[60] X. Lu, A. Moffat, and J. S. Culpepper. On the cost of extracting proximity features for term-dependency models. In *Proc. Int. Conf. Information and Knowledge Management*, pages 293–302, 2015.

[61] X. Lu, A. Moffat, and J. S. Culpepper. Modeling relevance as a function of retrieval rank. In *Proc. Asian Information Retrieval Societies Conf.*, pages 3–15, 2016.

[62] X. Lu, A. Moffat, and J. S. Culpepper. Efficient and effective higher order proximity modeling. In *Proc. Int. Conf. Theory of Information Retrieval*, pages 21–30, 2016.

[63] X. Lu, A. Moffat, and J. S. Culpepper. The effect of pooling and evaluation depth on IR metrics. *Information Retrieval J.*, 19(4):416–445, 2016.

[64] X. Lu, A. Moffat, and J. S. Culpepper. Can deep effectiveness metrics be evaluated using shallow judgment pools? In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 35–44, 2017.

[65] Y. Lv and C. Zhai. Positional language models for information retrieval. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 299–306, 2009.

[66] C. Macdonald and I. Ounis. Global statistics in proximity weighting models. In *Proc. ACM-SIGIR Wrkshop. on Web N-gram*, pages 30–37, 2010.

[67] C. Macdonald, I. Ounis, and N. Tonellotto. Upper-bound approximations for dynamic pruning. *ACM Trans. Information Systems*, 29(4):17, 2011.

[68] C. Macdonald, R. L. T. Santos, and I. Ounis. On the usefulness of query features for learning to rank. In *Proc. Int. Conf. Information and Knowledge Management*, pages 2559–2562, 2012.

[69] C. Macdonald, R. L. T. Santos, I. Ounis, and B. He. About learning models with multiple query-dependent features. *ACM Trans. Information Systems*, 31(3):11, 2013.

[70] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[71] Q. Mei, H. Fang, and C. Zhai. A study of Poisson query generation model for information retrieval. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 319–326, 2007.

[72] D. Metzler. *Beyond bags of words: Effectively modeling dependence and features in information retrieval*. PhD thesis, University of Massachusetts Amherst, 2007.

[73] D. Metzler and W. B. Croft. A Markov random field model for term dependencies. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 472–479, 2005.

[74] A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Trans. Information Systems*, 27(1), 2008.

[75] A. Moffat, W. Webber, and J. Zobel. Strategic system comparisons via targeted relevance judgments. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 375–382, 2007.

[76] A. Moffat, P. Bailey, F. Scholer, and P. Thomas. INST: An adaptive metric for information retrieval evaluation. In *Proc. Australasian Doc. Comp. Symp.*, page 5, 2015.

[77] G. Navarro and V. Mäkinen. Compressed full-text indexes. *ACM Computing Surveys*, 39 (1):2, 2007.

[78] J. Peng, C. Macdonald, B. He, V. Plachouras, and I. Ounis. Incorporating term dependency in the DFR framework. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 843–844, 2007.

[79] M. Petri and A. Moffat. On the cost of phrase-based ranking. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 931–934, 2015.

[80] M. Petri, A. Moffat, and J. S. Culpepper. Score-safe term-dependency processing with hybrid indexes. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 899–902, 2014.

[81] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 275–281, 1998.

[82] Y. Rasolofo and J. Savoy. Term proximity scoring for keyword-based retrieval systems. In *Proc. European Conf. on Information Retrieval*, pages 207–218, 2003.

[83] S. D. Ravana and A. Moffat. Score estimation, incomplete judgments, and significance testing in IR evaluation. In *Proc. Asian Information Retrieval Societies Conf.*, pages 97–109, 2010.

[84] S. E. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4): 294–304, 1977.

[85] S. E. Robertson. On GMAP: And other transformations. In *Proc. Int. Conf. Information and Knowledge Management*, pages 78–83, 2006.

[86] S. E. Robertson. A new interpretation of average precision. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 689–690, 2008.

[87] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *J. of the Am. Soc. for Information Science*, 27(3):129–146, 1976.

[88] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[89] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. Text REtrieval Conf.*, pages 109–126, 1995.

[90] S. E. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *Proc. Int. Conf. Information and Knowledge Management*, pages 42–49, 2004.

[91] S. E. Robertson, E. Kanoulas, and E. Yilmaz. Extending average precision to graded relevance judgments. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 603–610, 2010.

[92] K. Sadakane and H. Imai. Text retrieval by using k-word proximity search. In *Proc. Database Applications in Non-Traditional Environments*, pages 183–188, 1999.

[93] T. Sakai. New performance metrics based on multigrade relevance: Their application to question answering. In *Proc. NII Testbeds and Communities for Information Access and Research*, 2004.

[94] T. Sakai. Evaluating evaluation metrics based on the bootstrap. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 525–532, 2006.

[95] T. Sakai. Alternatives to BPref. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 71–78, 2007.

[96] T. Sakai. Comparing metrics across TREC and NTCIR: The robustness to system bias. In *Proc. Int. Conf. Information and Knowledge Management*, pages 581–590, 2008.

[97] T. Sakai. Comparing metrics across TREC and NTCIR: The robustness to pool depth bias. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 691–692, 2008.

[98] T. Sakai. Metrics, statistics, tests. In *Proc. PROMISE Winter School*, pages 116–163, 2014.

[99] T. Sakai. Topic set size design. *Information Retrieval J.*, 19(3):256–283, 2016.

[100] T. Sakai. Two sample t-tests for IR evaluation: Student or Welch? In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 1045–1048, 2016.

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

[102] M. Sanderson. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval*, 4(4):247–375, 2010.

[103] M. Sanderson and W. B. Croft. The history of information retrieval research. *Proceedings of the IEEE*, 100(Special Centennial Issue):1444–1451, 2012.

[104] M. Sanderson and J. Zobel. Information retrieval system evaluation: Effort, sensitivity, and reliability. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 162–169, 2005.

[105] T. Schnabel, A. Swaminathan, P. Frazier, and T. Joachims. Unbiased comparative evaluation of ranking functions. In *Proc. Int. Conf. Theory of Information Retrieval*, pages 109–118, 2016.

[106] A. Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4): 35–43, 2001.

[107] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proc. Int. Conf. Information and Knowledge Management*, pages 623–632, 2007.

[108] F. Song and W. B. Croft. A general language model for information retrieval. In *Proc. Int. Conf. Information and Knowledge Management*, pages 316–321, 1999.

[109] R. Song, M. J. Taylor, J.-R. Wen, H.-W. Hon, and Y. Yu. Viewing term proximity from a different perspective. In *Proc. European Conf. on Information Retrieval*, pages 346–357, 2008.

[110] M. Srikanth and R. Srihari. Biterm language models for document retrieval. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 425–426, 2002.

[111] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 295–302, 2007.

[112] M. Taube, C. D. Gull, and I. S. Wachtel. Unit terms in coordinate indexing. *J. of the Soc. for Information Science and Technology*, 3(4):213–218, 1952.

[113] A. Trotman, A. Puurula, and B. Burgess. Improvements to BM25 and language models examined. In *Proc. Australasian Doc. Comp. Symp.*, pages 58–65, 2014.

[114] H. Turtle and W. B. Croft. Inference networks for document retrieval. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 1–24, 1989.

[115] E. M. Voorhees. The philosophy of information retrieval evaluation. In *Proc. CLEF Wrkshp. European languages*, pages 355–370, 2001.

[116] E. M. Voorhees. Overview of TREC 2004. In *Proc. Text REtrieval Conf.*, pages 1–13, 2003.

[117] E. M. Voorhees. The effect of sampling strategy on inferred measures. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 1119–1122, 2014.

[118] E. M. Voorhees and D. K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. The MIT Press, 2005.

[119] J. B. P. Vuurens and A. P. de Vries. Distance matters! Cumulative proximity expansions for ranking documents. *Information Retrieval J.*, 17(4):380–406, 2014.

[120] L. Wang, J. Lin, and D. Metzler. A cascade ranking model for efficient ranked retrieval. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 105–114, 2011.

[121] W. Webber and L. A. F. Park. Score adjustment for correction of pooling bias. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 444–451, 2009.

[122] W. Webber, A. Moffat, and J. Zobel. Statistical power in retrieval experimentation. In *Proc. Int. Conf. Information and Knowledge Management*, pages 571–580, 2008.

[123] W. Webber, A. Moffat, and J. Zobel. The effect of pooling and evaluation depth on metric stability. In *Proc. Wrkshp. Evaluation Information Access*, pages 7–15, 2010.

[124] W. Webber, A. Moffat, and J. Zobel. A similarity measure for indefinite rankings. *ACM Trans. Information Systems*, 28(4):20:1–20:38, 2010.

[125] H. E. Williams, J. Zobel, and D. Bahle. Fast phrase querying with combined indexes. *ACM Trans. Information Systems*, 22(4):573–594, 2004.

[126] Z. Yang, A. Moffat, and A. Turpin. How precise does document scoring need to be? In *Proc. Asian Information Retrieval Societies Conf.*, pages 279–291, 2016.

[127] E. Yilmaz and J. A. Aslam. Estimating average precision with incomplete and imperfect judgments. In *Proc. Int. Conf. Information and Knowledge Management*, pages 102–111, 2006.

[128] E. Yilmaz and J. A. Aslam. Estimating average precision when judgments are incomplete. *Knowl. Inf. Syst.*, 16(2):173–211, 2007.

[129] E. Yilmaz, J. A. Aslam, and S. E. Robertson. A new rank correlation coefficient for information retrieval. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 587–594, 2008.

[130] E. Yilmaz, E. Kanoulas, and J. A. Aslam. A simple and efficient sampling method for estimating AP and NDCG. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 603–610, 2008.

[131] C. Zhai. Statistical language models for information retrieval. *Foundations and Trends in Information Retrieval*, 1(1):1–141, 2008.

[132] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 334–342, 2001.

[133] Z. Zhou. *Ensemble Methods: Foundations and Algorithms*. CRC press, 2012.

[134] J. Zobel. How reliable are the results of large-scale information retrieval experiments? In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 307–314, 1998.

[135] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38 (2):6:1–6:56, 2006.