Automatic Methods for Low-Cost Evaluation and Position-Aware Models for Neural Information Retrieval

A dissertation submitted towards the degree

Doctor of Natural Science/Engineering

of the Faculty of Mathematics and Computer Science of

Saarland University

by

Kai Hui

Saarbrücken
2017

Colloquium

Date	:	04.12.2017
Place	:	Saarbrücken
Dean	:	Prof. Frank-Olaf Schreyer

Examination Board

Chairman	:	Prof. Dietrich Klakow
Supervisor and Reviewer	:	Prof. Klaus Berberich
Reviewer	:	Prof. Gerhard Weikum
Reviewer	:	Prof. Laura Dietz
Scentific Assitant	:	Dr. Andrew Yates

Abstract

An information retrieval (IR) system assists people in consuming huge amount of data, where the evaluation and the construction of such systems are important. However, there exist two difficulties: the overwhelmingly *large number of query-document pairs* to judge, making IR evaluation a manually laborious task; and the *complicated patterns* to model due to the non-symmetric, heterogeneous relationships between a query-document pair, where different interaction patterns such as term dependency and proximity have been demonstrated to be useful, yet are non-trivial for a single IR model to encode. In this thesis we attempt to address both difficulties from the perspectives of IR evaluation and of the retrieval model respectively, by reducing the manual cost with automatic methods, by investigating the usage of crowdsourcing in collecting preference judgments, and by proposing novel neural retrieval models.

In particular, to address the *large number of query-document pairs* in IR evaluation, a low-cost *selective labeling* method is proposed to pick out a small subset of representative documents for manual judgments in favor of the follow-up prediction for the remaining query-document pairs; furthermore, a language-model based cascade measure framework is developed to evaluate the novelty and diversity, utilizing the content of the labeled documents to *mitigate incomplete labels*. In addition, we also attempt to make the preference judgments practically usable by empirically investigating different properties of the judgments when collected via crowdsourcing; and by proposing a novel judgment mechanism, making a compromise between the judgment quality and the number of judgments.

Finally, to model different *complicated patterns* in a single retrieval model, inspired by the recent advances in deep learning, we develop novel neural IR models to incorporate different patterns like term dependency, query proximity, density of relevance, and query coverage in a single model. We demonstrate their superior performances through evaluations on different datasets.

Kurzfassung

Ein Information-Retrieval (IR) System hilft Menschen bei der Arbeit mit großen Datenmengen, daher ist die Entwicklung und Evaluation solcher Systeme wichtig. Allerdings gibt es zwei Herausforderungen: die große Anzahl von Anfrage-Dokument-Paaren, die manuelle IR-Evaluation schwierig macht; sowie die komplizierten zu modellierenden Muster, aufgrund der nicht-symmetrischen, heterogenen Beziehung zwischen einem Anfragen und Dokumenten, wo erwiesen ist dass verschiedene Interaktionsmuster wie Termabhängigkeiten und Termnähe wichtig sind, aber nicht einfach durch ein einzelnes IR-Modell zu erfassen sind. In dieser Dissertation versuchen wir, beide Herausforderungen aus der Perspektive der IR-Evaluation bzw. der IR-Modellierung anzugehen, indem wir die manuellen Kosten mit automatischen Methoden reduzieren, indem wir die Verwendung von Crowdsourcing bei der Erfassung von Präferenzbewertungen untersuchen und indem wir neue neuronale IR-Modelle vorschlagen.

Um die große Anzahl von Anfrage-Dokument-Paaren in der IR-Evaluation in Angriff zu nehmen, schlagen wir eine kostengünstige selektive Bewertungsmethode vor, die nur eine kleine Untermenge von repräsentativen Dokumenten für manuelle Beurteilungen auswählt, deren Ergebnisse dann extrapoliert werden; darüber hinaus wird ein unüberwachtes sprachmodellbasiertes Gütemaß für Neuheit und Diversität vorgeschlagen, wobei der Inhalt der bewerteten Dokumente genutzt wird, um unvollständige Bewertungen zu kompensieren. Außerdem versuchen wir Präferenzbewertungen praktisch nutzbar zu machen, indem wir empirisch verschiedene Eigenschaften der Bewertungen beim Sammeln über Crowdsourcing untersuchen, und indem wir einen neuartigen Bewertungsmechanismus entwickeln, der einen Kompromiss zwischen der Bewertungsqualität und der Anzahl der Bewertungen macht.

Abschließend, um verschiedene komplizierte Muster in einem einzigen IR-Modell zu erfassen, inspiriert von den jüngsten Fortschritten bei Deep-Learning-Verfahren, entwickeln wir neuartige neuronale IR-Modelle, die verschiedene Muster wie Termabhängigkeit, Termnähe, Relevanzdichte sowie Anfrageabdeckung in einem einzelnen IR-Modell integrieren. Experimente auf verschiedenen Datensätzen zeigen die überlegene Performance des vorgeschlagenen IR-Modells.

Dedicated to my parents

Acknowledgments

I would thank my supervisor Prof. Klaus Berberich for all his helps during my doctoral study. It is a great luck for me to have a supervisor like Klaus, who is conscientious, responsible and very accessible during the whole procedure. He helps me a lot in determining the research direction, in writing good research papers and in presenting research works. His feedbacks are also important when my writing this thesis. I would also like to thank Prof. Laura Dietz and Prof. Gerhard Weikum for agreeing to review my thesis. I especially want to express my gratefulness for Prof. Gerhard Weikum and the International Max Planck Research School for supporting me finishing my doctoral research. In addition, I am glad to get the chances to meet and work together with all my colleagues at D5 group, where we enjoy an exciting workplace. Especially, I would like to thank my co-author Dr. Andrew Yates, with whom we have a successful and enjoyable corporation.

Last, but surely not least, I would like to thank my parents for their raising, supports and encouragements all along the way. They are the strongest motivations for my every single progress. Also, I would like to thank my girlfriend Yuexin Cao for her company in the past years.

Contents

1 Introduction

An information retrieval (IR) system is a crucial instrument when consuming huge amount of data. Evaluating such systems and building an effective retrieval model are two important problems. On one hand, to measure how well the IR system can satisfy users' information needs, manual judgments are employed to provide a ground-truth ranking for the queries, and the comparison between this ground-truth ranking and the ranking provided by an IR system is used to measure its performance. Therefore, an assessor needs to judge all query-document pairs to create a ground-truth ranking. On the other hand, an IR system aims at filtering the information according to a provided information need, reducing the workload in follow-up information processing steps, enabling an user to focus on a small subset of relevant information. In this procedure, a retrieval system returns documents in response to an information request (a.k.a., a *topic* or a *query*) from an user and ranks the huge amount of documents in descending order according to their relevance (a.k.a., the *search results*), making the user focus on few top-ranked documents. To do so, an IR system needs to model all kinds of interaction signals between individual query-document pairs, determining the relevance degree accordingly.

There exist two difficulties, however, namely, the *large number of query-document pairs* and the *complicated patterns* involved in the interaction. In particular, the concept of relevance always involves both queries and documents, making the number of pairs depend on both of them, leading to a large number of query-document combinations, which are especially overwhelming for human assessors, making IR evaluation a manually laborious task. Meanwhile as a result of the non-symmetric, heterogeneous relationship between a query and a document, capturing all kinds of interaction patterns like term dependency and query proximity in a single retrieval model becomes a challenging task.

In this thesis we attempt to address these two difficulties from the perspectives of IR evaluation and of the retrieval model respectively, by reducing the manual cost with automatic methods, by investigating the usage of crowdsourcing in collecting the preference judgments, and by proposing novel retrieval models powered by deep learning. The sketch of the research problems and our contributions in each chapter are listed in the following:

- To address the expensiveness of manual judgments, recent works seek to reduce the manual cost in IR evaluation by *selectively labeling* a subset of query-document pairs as in (Carterette et al., 2006) and by *mitigating incomplete labels* with different methods as

in (Buckley and Voorhees, 2004), where, however, the incorporation of machine learning methods in either phase has been insufficiently investigated.
In Chapter 3, we propose two different methods to reduce manual judgments in favor of evaluation for ad-hoc retrieval and for novelty and diversity separately. In particular, a low-cost *selective labeling* method named MAXREP (Hui and Berberich, 2015) is proposed to pick out a small subset of representative documents for manual judgments, together with which different document representations are also investigated (Hui and Berberich, 2016); in addition, a cascade measure framework, named LMD-CASCADE (Hui et al., 2017a), is proposed to evaluate the novelty and diversification, utilizing the content of the labeled documents to *mitigate incomplete labels*.

- *Preference judgments*, one particular kind of manual judgment, has been demonstrated to be a better alternative to collect manual assessments more accurately (Carterette et al., 2008), which, nevertheless, suffer even seriously from the large number of judgments. Meanwhile, crowdsourcing has been demonstrated with a potential to dramatically reduce the manual cost in the judgment procedure (Alonso and Mizzaro, 2009; Kazai et al., 2013). Thereby, one natural thought would be to combine these two, achieving superior quality of judgments with cheap cost at the same time, which has not been fully investigated.
  In Chapter 4, we first reexamine the number of preference judgments and empirically investigate the number of judgments by simulating the ground-truth rankings, ultimately highlighting a novel way to reduce the number of preference judgments (Hui and Berberich, 2017a,b). Moreover, we empirically investigate the transitivity, time consumption, and judgment quality when collecting two kinds of preference judgments via crowdsourcing in place of traditional centralized expert-dependent labeling paradigm (Hui and Berberich, 2017c), demonstrating that the combination of preference judgments and crowdsourcing could lead to a superior judgment quality with a reduced number of judgments by assuming transitivity.
- Decades of research on ad-hoc IR highlighted multiple useful interaction patterns which could be potentially used to enhance the performance of a retrieval model; meanwhile the recent advances of deep learning provides a variety of building blocks with strength to model all kinds of patterns (Goodfellow et al., 2016), making the incorporation of different patterns in a single model possible.
  As the third part, in Chapter 5, inspired by the importance of positional interactions from the literature, we develop novel neural IR models named PACRR (Hui et al., 2017b,c) and RE-PACRR (Hui et al., 2018)[1], incorporating different patterns like term dependency, query proximity, density of relevance, and query coverage. Through intensive experiments on established benchmarks, we demonstrate that the proposed models can significantly advance the state-of-the-art retrieval models.

[1] https://arxiv.org/abs/1706.10192v2

2 Background

2.1 Ranking Models in Information Retrieval

In this chapter, several classical retrieval models are briefly introduced. In the decades of research in information retrieval, such models aim at providing answers to the same research question, namely, “how to sort the documents better relative to a given query”. In this procedure, different ad-hoc retrieval models are proposed, thereafter, the novelty and diversity are further considered to improve the search results. In Section 2.1.1, the probability ranking principle (*PRP*) is revisited, and the *tf-idf*, BM25 and the query-likelihood model are described with a running example. Furthermore, in Section 2.1.3, the ideas of novelty and diversity are summarized.

2.1.1 Ad-hoc Information Retrieval

When discovering and digesting information, one may face a dilemma between the quasi-infinite amount of available data and the limited ability to process them. An information retrieval system is designed to assist in this process, making humans focus on the most valuable information. A retrieval system returns documents in response to an information request from users and ranks them in descending order of their degree of relevance. Such information requests are referred to as *topics* or *queries*, and returned document lists are refereed to as *search results*.

Specifically, Robertson assumed that the relevance of a document is independent of the other documents in ad-hoc retrieval, and demonstrated that the probability ranking principle (*PRP*) can lead to optimal retrieval performance (Robertson, 1977), that is, a retrieval system should rank the documents in order of decreasing probability of relevance relative to the users’ query. Following *PRP*, different ad-hoc retrieval models have been developed to weight a query-document pair to quantify the relevance of the document. The relevance weights are computed by combining features from different perspectives, such as the quality of a document, and the occurrences of query terms in a document.

One simple weighting model is the *tf-idf* model, proposed in the context of the vector space model (Manning et al., 2008). *Tf-idf* is short for term frequency-inverse document frequency, only considering two features, namely, the occurrence of query terms and the importance of these terms. The weighting model of *tf-idf* is summarized in Equation 2.1. The term frequency represents the number of occurrences of a query term in the evaluated document, meanwhile the $idf(t) = log\frac{N}{df(t)}$, where N is the total number of documents in the collection and df represents the number of documents which include query term t (Robertson, 2004). Given a query and a list of documents, one computes *tf-idf* as in Equation 2.1 for each document, and ranks these documents in descending order of their tf-idf scores. Since *tf-idf*, different weighting models were proposed to advance the state-of-the-art, either by digesting the statistics in a more fine-grained manner or by incorporating more features. One example for the former situation is the BM25 (Best Match 25) model as summarized in Equation 2.2, which is motivated and derived from the probabilistic 2-Poisson model (Robertson and Walker, 1994). It can be seen that a document normalization factor is considered in computing the term frequency in $tf'(t_j, d_i)$, and a component to reward query terms that occur multiple times in the query is added. For the latter situation, Metzler and Croft incorporated term dependency beyond uni-gram matching (Metzler and Croft, 2005) meanwhile Tao and Zhai utilized the proximity of query term occurrences (Tao and Zhai, 2007).

$$tf\text{-}idf(q_i, d_i) = \sum_{t_j \in q_i} tf(t_j, d_i) idf(t_j) \tag{2.1}$$

$$\begin{gathered}
BM25(q_i, d_i) = \sum_{t_j \in q_i} tf'(t_j, d_i) idf(t_j) \frac{(k_3 + 1) tf(t_j, q_i)}{k_3 + tf(t_j, q_i)} \\
\text{where } tf'(t_j, d_i) = \frac{(k_1 + 1) tf(t_j, d_i)}{k_1((1-b) + b\frac{dl_{d_j}}{dl_{avg}}) + tf(t_j, d_i)} \\
dl_{.} \text{ is document length.}
\end{gathered} \tag{2.2}$$

An example. We demonstrate an example as follows to further illustrate this ranking procedure, using *tf-idf* as weighting models to keep it simple. Given a query “ad-hoc information retrieval model”, assume there are three documents for ranking. We demonstrate the computation in Table 2.1, and thereby the three documents can be sorted as follows accordingly: Document 3 $\succ$ Document 2 $\succ$ Document 1. From Table 2.1, it can be seen that the importance of different query term occurrences is also considered beyond the occurrences of query terms (*tf*). As an extreme example, the *idf* scores of “information” and “retrieval” are both 0, indicating that they contribute nothing in discriminating these three documents.

		ad-hoc	information	retrieval	model
N		3			
$df(t)$		2	3	3	1
$idf = log\frac{N}{df}$		$log\frac{3}{2}$	0	0	$log3$
$tf(t,d)$	# 1	0	1	1	0
	# 2	1	1	1	0
	# 3	1	1	1	1
$tf\text{-}idf = \sum_{t \in q} tf(t,d)idf(t)$	# 1	$0+0+0+0=0$			
	# 2	$log\frac{3}{2}+0+0+0=log\frac{3}{2}$			
	# 3	$log\frac{3}{2}+0+0+log3=log\frac{9}{2}$			

Table 2.1: A running example to compute *tf-idf*.

Query: ad-hoc information retrieval model.
Document 1: An *information retrieval model* for diversification and novelty.
Document 2: A comparative study of pseudo relevance feedback for *ad-hoc information retrieval*.
Document 3: A deep relevance matching *model* for *ad-hoc information retrieval*.

2.1.2 Language Model

Statistical language models have ample applications in natural language processing including tasks such as speech recognition and machine translation. More recently, within the last two decades, they have also been successfully applied to Information Retrieval tasks.

The language modeling approach was first introduced to Information Retrieval in (Ponte and Croft, 1998), where they proposed the query likelihood scoring method. The documents are ranked by the likelihood of a query according to the estimated document language model (Zhai, 2008a), where Θ_d is the language model of the document d. It is clear that the crucial component of this scoring function is the estimation of the language model of a document Θ_d. In (Ponte and Croft, 1998), the document language model is estimated with multiple Bernoulli distribution over an individual term q_t, namely, $P(q_t|\Theta_d) = \frac{tf(t,d)}{|d|}$. In the meantime, an unigram language model is employed, resulting in different query terms being independent with each other as indicated in Equation 2.3.

$$LM(q,d) = P(q|\Theta_d) = \prod_{q_t \in q} P(q_t|\Theta_d) = \prod_{q_t \in q} \frac{tf(t,d)}{|d|} \tag{2.3}$$

Unigram language models with Dirichlet smoothing. It is critical for the estimation of Θ_d to smooth the maximum likelihood where an unseen word is assigned zero probability. For different smoothing approaches, one could refer to (Zhai and Lafferty, 2001), which empirically compared different smoothing methods, including Jelinek-Mercer smoothing (Jelinek, 1980) and Dirichlet prior (MacKay and Peto, 1995) on several standard test collections.

The Dirichlet prior smoothing method can be derived by using Bayesian estimation with a Dirichlet conjugate prior, as indicated in (MacKay and Peto, 1995) and (Zhai and Lafferty, 2002). The formula is summarized in Equation 2.4, where $P(q_t|\Theta_{\mathcal{D}})$ is a collection language model, as indicated in Equation 2.5, and μ is a tunable parameter which controls the influence of Dirichlet smoothing in $\Theta_{\mathcal{D}}$. The smoothing with the document collection language model $\Theta_{\mathcal{D}}$ can be interpreted as the prior knowledge of an user about general documents from the collection when going through the search results.

$$P(q_t|\Theta_d) = \frac{tf(q_t, d) + \mu}{|d| + \mu P(q_t|\Theta_{\mathcal{D}})} . \tag{2.4}$$

$$P(q_t|\Theta_{\mathcal{D}}) = \frac{\sum_{d \in \mathcal{D}} tf(q_t, d)}{\sum_{d \in \mathcal{D}} |d|} . \tag{2.5}$$

2.1.3 Novelty and Diversity

As mentioned, in ad-hoc retrieval, one aims at ranking documents according to their relevance and different documents are deemed independent from one another. Beyond ad-hoc retrieval, Carbonell and Goldstein further considered the dependency among documents, namely, the relevance of a document also depends on documents that are ranked higher in the same search results (Carbonell and Goldstein, 1998). Intuitively, when going through a search result, an user may find a duplicated document no longer useful. Moreover, given an ambiguous query, which includes different interpretations, also known as *aspect* or *subtopic*, a good search result should cover more different *aspects* to meet different intents. One example is an ambiguous query from TREC Web Track 2013: “rain man”[1]. In TREC, the query “rain man” is interpreted in five different ways as follows.

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

Query rain man.

Subtopic 1. Where can I watch the full "Rain Main" movie online?
Subtopic 2. Find information about the real person on which the Rain Man movie is based.
Subtopic 3. Find movie reviews of "Rain Man".
Subtopic 4. Find quotes from the "Rain Man" movie.
Subtopic 5. Find the lyrics to Eminem's "Rain Man".

2.2 Evaluation in Information Retrieval

Evaluation in information retrieval is to compare the performance of different rivaling retrieval systems, comparing how well the different systems can satisfy users' information needs. The research in IR evaluation aims to measure, compare and analyze different retrieval methods, with an attempt for a better understanding of the retrieval problems. In this chapter, the evaluation paradigm and a widely employed benchmark are introduced in Section 2.2.1; thereafter, different assessors and different kinds of judgments are summarized in Section 2.2.2 and Section 2.2.3 respectively. Finally, several commonly-used evaluation measures are reviewed in Section 2.2.4.

2.2.1 Cranfield Paradigm and TREC Benchmark

Evaluation in information retrieval aims at comparing performances among different retrieval systems. Off-line evaluation in information retrieval often relies on the Cranfield paradigm (Cleverdon, 1991), which introduced formal testing of IR systems on test databases in a controlled and laboratory-like setting (Manning et al., 2008). Specifically, to compare the performance of several information retrieval systems, one agrees on a set of information requests (also called *topics* or *queries*), which are representative of the target workload. Each of these information requests is then formulated as a keyword query, and results are obtained from each of the information retrieval systems under comparison. Following that, human assessors label retrieved result documents with regard to their relevance. These manual judgments determine a ground-truth ranking over all retrieved documents. Search results from rivaling systems are examined and compared in terms of their agreements with the ground-truth ranking. This comparison is conducted by computing different kinds of evaluation measures, such as nDCG (Järvelin and Kekäläinen, 2002) and ERR-IA (Chapelle et al., 2011), which are introduced in Section 2.2.4. The evaluation pipeline is visualized in Figure 2.1.

Best known examples following Cranfield paradigm are test collections from TREC, short for Text Retrieval Evaluation Conference, which is held by the U.S. National Institute of Standards and Technology (NIST) since 1992. In this work, the test collections from TREC Web Track

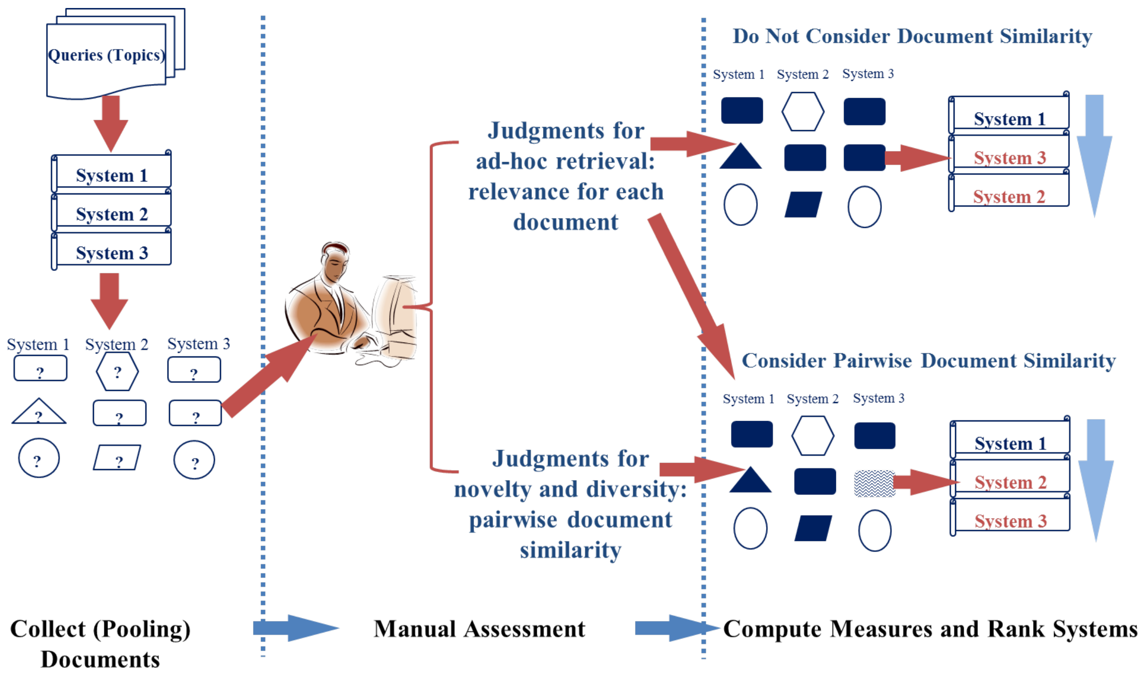

Queries (Topics)
System 1
System 2
System 3
System 1 System 2 System 3
Judgments for ad-hoc retrieval: relevance for each document
Judgments for novelty and diversity: pairwise document similarity
Do Not Consider Document Similarity
System 1 System 2 System 3
System 1
System 3
System 2
Consider Pairwise Document Similarity
System 1 System 2 System 3
System 1
System 2
System 3
Collect (Pooling) Documents
Manual Assessment
Compute Measures and Rank Systems

Figure 2.1: The pipeline of off-line evaluation in information retrieval. From the left to the right, one first collects documents from the search results returned by rivaling systems; thereafter assessors are hired to make judgments; and these judgments are consumed to compute different evaluation measures. The rivaling systems are sorted according to the measure scores.

also serve as major benchmarks. We employ the evaluation procedure in TREC Web Track to further interpret the evaluation pipeline. NIST created and assessed 50 topics for each year of TREC Web Track. For the purposes of the diversity task, each topic is further structured into a list of subtopics, as the example query "Rain Man" from Section 2.1.3, related to different user needs. In the manual judgments procedure, a rivaling system in the ad-hoc and diversity tasks submits a ranking containing top 1000 or top 10,000 documents for a given topic. Documents that are within top-k, e.g., top-20, from different submissions are pooled for manual judgments. Afterwards, documents in this common pool, together with the definition of the topic are assessed by trained assessors from NIST. Finally, the search results from each rivaling system are converted to a real-valued score, according to certain evaluation measures, and different systems are ranked according to these measure scores (Collins-Thompson et al., 2015a).

2.2.2 Trained Assessors and Crowdsourcing

As mentioned, the manual judgment procedure is conducted by trained assessors in TREC Web Track, who are deemed to be reliable and accurate. As argued in (Alonso and Mizzaro, 2012), however, the assessment requires resources in terms of infrastructure, organization, time, money, and does not scale up easily. This motivated many works on low-cost evaluation, and this thesis is one of them. Given the difficulty and the high cost to hire trained assessors, crowdsourcing has been known as a good alternative to collect manual judgments.

Howe stated that "Crowdsourcing is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call" (Howe, 2008). Crowdsourcing platforms such as Amazon Mechanical Turk[2] and Crowdflower[3] have provided a way to reach out to a large crowd of diverse workers for judgments, which is known as inexpensive and scalable (Alonso and Baeza-Yates, 2011). Thereby, the differences between the trained assessors and the crowdsourcing are that the latter relies on more diversified manual labors instead of few more professional assessors in-house. The details about configuration of a crowdsourcing task are described in Section 4.4.1.

2.2.3 Preference and Graded Judgments

As demonstrated in Figure 2.2, there exist two approaches to collect judgments, namely, graded judgments, where documents are labeled independently with a predefined graded level, and preference judgments, where judges provide a relative ranking for a pair of documents. Though, the collected judgments are in different forms, both lead to a ground-truth document ranking. For instance, given a test query, assuming there are two rivaling systems s_1 and s_2 and their search results in response to the test query are as follows.

$$
\begin{aligned}
s1 &: d_3, d_1, d_2 \\
s2 &: d_5, d_4, d_2, d_1
\end{aligned}
$$

To compare these two search results, manual judgments are collected in terms of either graded judgments or preference judgments. When using graded judgments, the five documents are assessed by judges independently and are assigned a predefined graded level, say $d_1 : 0$, $d_2 : 1$, $d_3 : 1$, $d_4 : 1$, $d_5 : 2$; when using preference judgments, pairwise preferences over document pairs are collected, say $d_5 \succ d_4$, $d_5 \succ d_3$, $d_5 \succ d_2$, $d_5 \succ d_1$, $d_4 \succ d_3$, $d_4 \succ d_2$, $d_4 \succ d_1$, $d_3 \succ d_2$, $d_3 \succ d_1$, $d_2 \succ d_1$. We use $\succ$, $\prec$ and $\sim$ to denote "better than", "worse than", and "tie" relationships. Ultimately, with both kinds of judgments, a ground-truth ranking of documents can be determined according to their relevance. For our example we obtain:

$$
\begin{aligned}
graded\ judgments &: d_5 \succ d_4 \sim d_3 \sim d_2 \succ d_1 \\
preference\ judgments &: d_5 \succ d_4 \succ d_3 \succ d_2 \succ d_1
\end{aligned}
$$

[2] https://www.mturk.com/mturk/
[3] https://www.crowdflower.com/

How well does the document A match the query?
- ☐ Highly-Relevant
- ☐ Relevant
- ☐ Non-Relevant

Which document is more relevant or they are equivalent to the query?
- ☐ A is more relevant
- ☐ A and B are equivalent
- ☐ B is more relevant

Figure 2.2: Examples for graded (left) and preference judgments (right).

2.2.4 Evaluation Measures

In this section, we introduce evaluation measures for ad-hoc retrieval, namely, *normalized discounted cumulative gain* (nDCG) (Järvelin and Kekäläinen, 2002), and *expected reciprocal rank* (ERR) (Chapelle et al., 2009). In addition, both of them are extended and used to evaluate novelty and diversity, namely, α-nDCG (Clarke et al., 2008) and ERR-IA (Chapelle et al., 2011). Beyond that, we also introduce another standard measure for diversification: *novelty- and rank-biased precision* (NRBP) from (Clarke et al., 2009).

Measures for Ad-hoc Information Retrieval

nDCG. Järvelin and Kekäläinen introduced nDCG to digest graded judgments beyond binary judgments. Moreover, nDCG discounts the position factors, devaluating documents that ranked lower in a ranking of documents. Additionally, nDCG introduces a relative-to-the-ideal component to normalize the measure score, removing the factors which come from queries instead of the evaluated ranking, making measure scores from different queries comparable. In particular, the cumulative gain at rank k is defined as

$$DCG_k = \sum_{i=1}^{k} \frac{gain(l_i)}{log(i+1)}$$

, where l_i represents the graded label of the document at rank i, and the $log(i+1)$ is a discount factor relative to the position. And the $gain(l_i)$ is a function mapping each graded level to a gain value. Two common functions are $gain(l_i) = l_i$ and $gain(l_i) = 2^{l_i} - 1$. Finally, to normalize the score, an ideal ranking is generated from all N_r relevant documents in the document collections. In the ideal ranking, documents are directly sorted by their graded levels. Thereafter, the DCG

is computed on this ideal ranking accordingly, which is denoted as $IDCG$. The formula is summarized in Equation 2.6.

$$nDCG_k = \frac{DCG_k}{IDCG_k} \tag{2.6}$$

ERR. Chapelle et al. proposed ERR, which is different from the assumption in nDCG that the relevance of a document is independent in the result ranking. Instead it captures the dependency among documents by assuming a cascade-style user model, including "diminishing returns" for redundant documents (Chapelle et al., 2009). Thus the contribution of each document is also based on the relevance of documents ranked above it. Intuitively, when browsing a search result, an user is very likely to stop early after getting enough relevant information before reaching the end of the search results. Thus, the discount function is not just dependent on the rank but also on relevance of previously ranked documents as in Equation 2.7. Here, R_i is a function of the relevance grade of the document appearing at position i in the ranking, and it is commonly defined as $(2^{l_i} - 1)/2^{l_{max}}$, where l_{max} is the highest graded level.

$$ERR_k = \sum_{i=1}^{k} \frac{1}{i} \prod_{j=1}^{i-1} (1 - R_j) \tag{2.7}$$

Measures for Diversity

While standard effectiveness measures evaluate IR systems only in terms of the relevance of returned results, more recently proposed measures also attempt to capture their diversity. Researchers proposed measures which take into account the diversity of returned results. These measures attempt to quantify the extent to which a given ranking minimizes the redundancy or maximizes the diversity of information provided to users. Extending ERR, Chapelle et al. proposed ERR-IA to further measure the diversification of the ranking, following the general approach in (Agrawal et al., 2009). Clarke et al. considered underspecified queries, namely, queries with faceted interpretations. They presented α-nDCG, which decomposes the information needs behind a query into so-called information nuggets and defines the utility of a document as the number of novel nuggets covered in a document. In a subsequent work they proposed NRBP (Clarke et al., 2009) which considers both ambiguous and underspecified (faceted) queries by combining α-nDCG and *Rank-Biased Precision* (RBP) proposed in (Moffat and Zobel, 2008).

α-nDCG. Clarke et al. extended the traditional nDCG (Järvelin and Kekäläinen, 2002) to α-nDCG metric (Equation 2.9) in evaluating diversity in search results. α-nDCG scores a result set by rewarding results relevant to new subtopics and penalizing the ones relevant to redundant subtopics. It balances relevance and diversity through the tuning of the α parameter. The larger the value of α, the more diversity is rewarded. We used $\alpha = 0.5$ for our experiments to give equal importance to relevance and diversity. Specifically, different from nDCG, where the gain reflects the graded relevance value of the document to the query, α-nDCG uses a *novelty-biased gain*, which is defined in Equation 2.8:

$$NG[r] = \sum_{i=1}^{m} J_i(r)(1-\alpha)^{C_i(r-1)} \tag{2.8}$$

$$\alpha\text{-}nDCG = \frac{\sum_{r=1}^{k} \frac{\sum_{i=1}^{m} J_i(r)(1-\alpha)^{C_i(r-1)}}{log_2(1+r)}}{IDCG} . \tag{2.9}$$

The $J_i(r)$ is a flag which indicates if the document at rank r is relevant or not to the intent i, and $C_i(r-1)$ is the number of times the subtopic i is covered by documents appearing before rank r. Discounting the value of each document covering the intent based on the number of documents that are already seen for it gives the *discounted cumulative gain* (α-DCG). Then, a normalization is performed to compare the scores against various topics. The normalization can be done by finding an “ideal” ranking that maximizes α-DCG as in nDCG. Since computing the ideal ranking is an NP-Complete problem (Carterette, 2011), a greedy algorithm can be used to compute an approximate solution. The ratio of α-DCG to this ranking gives α-nDCG.

ERR-IA is the intent-aware version of the *Expected Reciprocal Rank* (ERR). It is defined as the weighted average of ERR computed separately for each query subtopic (Chapelle et al., 2011), as summarized in Equation 2.10

$$ERR\text{-}IA = \sum_{i=1}^{m} p_i \sum_{r=1}^{k} \frac{1}{r} R_i(r) \prod_{j=1}^{r-1} (1 - R_i(j)) , \tag{2.10}$$

NRBP. The *Novelty- and Rank-Biased Precision* was proposed in (Clarke et al., 2009) to combine α-nDCG and RBP (*Rank Biased Precision*) (Moffat and Zobel, 2008). It is computed as in Equation 2.11.

$$NRBP = \frac{1-(1-\alpha)\beta}{m} \sum_{r=1}^{\infty} \beta^{r-1} \sum_{i=1}^{m} J_i(r)(1-\alpha)^{C_i(r)} . \tag{2.11}$$

As we can see, this metric uses two discount mechanisms: one is for the redundancy of documents and is based on the parameter α, the other one is based on the *persistence parameter* β, which is the probability that the user will go down in the ranked list of results. For our experiments in this work we set α and β as 0.5, following the default configuration in TREC[4].

[4] http://trec.nist.gov/data/web/12/ndeval.c

Rank Correlation

In addition to the ranking measure, we also employ Kendall's τ as a correlation measure between system rankings determined by different measures, serving as a major instrument to evaluate different evaluation measures in Chapters 3 and 4. Kendall's τ is the difference between concordant and discordant pairs divided by the total number of pairs. It ranges in $[-1, 1]$ with 1 (-1) indicating perfect agreement (disagreement).

2.3 Automatic Methods for Information Retrieval

Intuitively, a model aims at learning the way how a task could be fulfilled by a human, through training on labeled data, thereafter automatically fulfilling the task on unknown data points. When training a model, the input data is first encoded as feature vectors, and are fed into the model, together with their ground-truth labels. When training a model, in each iteration, a model conducts the task, e.g., prediction, by mapping the feature vectors of each input into an output variable, which is denoted as $\hat{y}$, based on the knowledge that the model has learned. Finally, the $\hat{y}$, together with the ground-truth label y, are fed into a loss function, to compare the output from the model against the provided ground-truth. The model learns from this loss functions, to mitigate the difference between its output and the provided ground-truth. This procedure is named supervised learning. One example for such loss function is the hinge loss, which is widely employed and is also the loss function employed in the support vector machine model, which is introduced in Section 2.3.2. As displayed in Equation 2.12, a model aims at adjusting the prediction where $y\hat{y} \leq 1$. For example, given a query q, a document d and its relevant label y. A relevance model, denoted as *rel*, aims at predicting the relevance of d relative to q. The output of the model can be written as $\hat{y} = rel(q, d; \Theta)$, where Θ represents the weights to be learned in the model. Therefore, the training procedure is to update Θ, minimizing the loss function in the form of $\mathcal{L}(y, \hat{y})$, making the model better mimic the provided ground-truth labels y.

In this section, we first review several methods which encode a document into the feature space. Thereafter, one of the mostly employed classification model, namely, the support vector machine (SVM), is described. Finally, we introduce a group of more powerful models, namely, neural networks, which include much more variables inside the model, leading to a superior performance on different applications.

$$\mathcal{L}(y, \hat{y}) = max(0, 1 - y\hat{y}) \tag{2.12}$$

2.3.1 Document Representations

Given that this thesis is mainly about ad-hoc retrieval, the relationships between a query and a document are of interests. Thus, we first review how to encode free text in a document or a query into the feature space. In particular, free text inputs like queries and documents are first converted to representations of words or documents before they could be digested by learning algorithms. In general, such representations attempt to be in favor of efficiency of the follow-up learning procedure, which highly depends on the dimensionality of the representation. Meanwhile, they also aim at preserving the semantic meaning of the text. For example, the relative distance among the representations of individual documents should preserve the relationship of their semantic meaning, e.g., documents which discuss similar topics or hold similar arguments should be close to each other. The document representations can be categorized into two classes, namely, the sparse representations, like the bag-of-words, and the dense representations, such as the vectors created with latent semantic analysis (Landauer et al., 1998), latent Dirichlet allocation (Blei et al., 2003), and the recently proposed PARA2VEC (Le and Mikolov, 2014).

In particular, since the choices of words can indicate the topic of a document, the bag-of-words sparse vector with tf-idf weighting is the default option in existing works like in (Büttcher et al., 2007). Without considering the order of terms, each document is represented as a sparse word vector, with components determined by tf-idf weighting as indicated in Equation 2.1. Since terms are treated as independent, their inter-relationship (e.g., synonymy) are neglected. To address this, the latent semantic analysis (LSA) represents the documents into a dense latent topic space by decomposing a document-term matrix. Thereafter, both documents and terms are represented with dense topic vectors. Likewise, Latent Dirichlet Allocation (LDA) (Blei et al., 2003) reduces the dimensionality with a generative model. In particular, in LDA, each document is assumed to be associated with a distribution of latent topics, and one has to first determine the topic distribution for individual documents before generating the term sequence. Meanwhile, individual terms are presumably sampled based on their topic, namely, from distribution determined by a $|Topic| \times |Vocabulary|$-dimension matrix. Therefore, the term sequence in a document could be generated with a two-phase sampling, namely, firstly sampling topics for the document, based on which the terms are drawn accordingly. More recently, document vectorization method based on neural network, such as PARA2VEC (Le and Mikolov, 2014), co-trains a document vector together with its word vectors, encoding the word co-occurrence information in this procedure. As a variant of the WORD2VEC method (Mikolov et al., 2013), PARA2VEC can be regarded as a neural network based method to encode the word embedding information from WORD2VEC into document representations. For more details, one could refer to the papers (Mikolov et al., 2013) and (Pennington et al., 2014) for word embeddings, and the listed references for different kinds of document representations.

2.3.2 Support Vector Machine

As one of the most popular discriminative model, support vector machine (SVM) has been widely employed as a classifier in different applications like text classification (Joachims, 1998). A SVM model is trained to adjust a hyperplane, by maximizing the margin between data points from either class in the training data. The optimization target is summarized in Equation 2.13, where $1/2||w||^2$ corresponds to a normalized margin, and ξ_i is a slack variable for $x^{(i)}$, allowing $x^{(i)}$ to be located inside the margin for $\xi_{(i)}$.

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \leq 1 - \xi_i,\ i = 1,\ldots,m \\ & \xi_i \leq 0,\ i = 1,\ldots,m. \end{aligned} \tag{2.13}$$

The training of a SVM model is equivalent to a quadratic optimization problem with constrains. One could either directly employ first-order or second-order optimization methods like gradient descent or Newton's method to directly solve this primal form in Equation 2.13, where w is the variable to be solved when minimizing the optimization target in the space of $x^{(i)}$. Alternatively, one could also utilize the Lagrange method to transform this primal problem into its dual problem, which is also a quadratic optimization problem. By solving the optimization in this dual form, one can directly get the weights for individual data points. The dual form is summarized in Equation 2.14, where the Lagrange multipliers $\alpha^{(i)}$ is the variable to be optimized on. The SMO (sequential minimal optimization) method proposed by Platt can efficiently solve this problem (Platt, 1998). The optimization in prime form can be solved with the Stochastic Gradient Descent method, making it very efficient to be solved; whereas the dual optimization is good for the utilization of kernel trick, namely, replacing the $x^T x$ in Equation 2.14 with a kernel function, and also utilizes the sparsity of the support vectors during inference. That being said, a test data point can be predicted by only comparing with the very few support vectors, instead of comparing against a hyperplane (w in the prime form).

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(\mathbf{i})^T}\mathbf{x}^{(j)} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C,\ i = 1,\ldots,m \\ & \sum_i \alpha_i y^{(i)} = 0. \end{aligned} \tag{2.14}$$

2.3.3 Neural Network

An Artificial Neural Network (ANN) is a model class that is motivated by the biological nervous systems. It is composed of a large number of highly interconnected processing units, which are known as neurons, and the weights are learned inside each neuron from the training data. Intuitively, a neural network utilizes the interconnections among different neurons, making a particular neuron to extract a small part of signals from the data flow. The internal computation in individual neurons, namely, the activation functions, and the way to organize different neurons, namely, the model structure, are the two most important aspects for the neural network. In this section, we start with the introduction of the neuron and some popular activation functions. Afterward, we summarize some basic architectures which normally serve as building blocks for a more complicated neural network.

Neurons and non-linear activations. A neuron is the basic unit in a neural network, which combines multiple inputs, thereafter feeding the combination into a non-linear activation. In Figure 2.3 a prototype of a neuron is displayed, where three inputs, namely, x_1, x_2 and x_3, are combined and are fed into the activation function f. The w_i and b_i are the weights to be learned, combining the three inputs into a scalar, thereafter, the f further applies on this scalar to introduce non-linearity into the model. Actually, parts of the strength from a deep model is due to this non-linearity posed by the activation functions in each of the thousand interconnected neurons, which make the model being able to learn more complicated patterns. Actually, an activation function is nothing but a function that maps a scalar to another scalar via some non-linear transformations, where the range of the output scalar varied. There are several activations that are widely employed, including sigmoid, tanh (hyperbolic tangent), and the ReLu (Rectifier Linear Unit). A sigmoid activation maps a scalar into the range $[0,1]$, normalizing the output to a probability, which can be seen from Equation 2.15. Whereas tanh is a centralized version of sigmoid by linear transformation, namely, $tanh(x) = 2\sigma(x) - 1$, mapping a scalar into $[-1,1]$, making the mean of the output equal zero. However, it is well-known that both sigmoid and tanh suffer from the vanishing gradient problem, as a result of the fact that the gradients are close to zero for most inputs.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.15}$$

To address this, ReLu (Nair and Hinton, 2010) has been proposed and widely used in recent deep models. Despite its simplicity as in Equation 2.16, ReLu addresses the vanishing gradient problem with a constant gradient for all input. Beyond that, ReLu introduces sparsity into the data flow, by turning half of the data into zero when $\sum w_i x_i + b_i \leq 0$, therefore improving the efficiency of training.

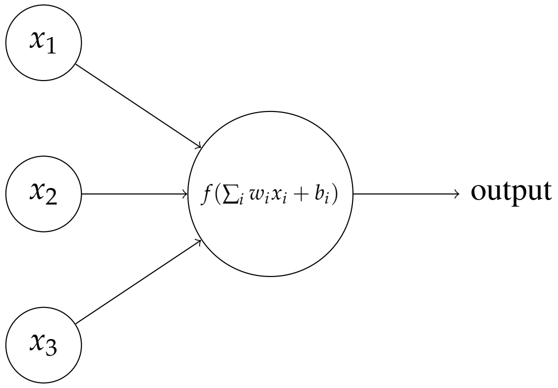


Figure 2.3: A single neuron with three inputs x_1, x_2 and x_3. f is an activation function to combine the three input signals and apply non-linear activation.

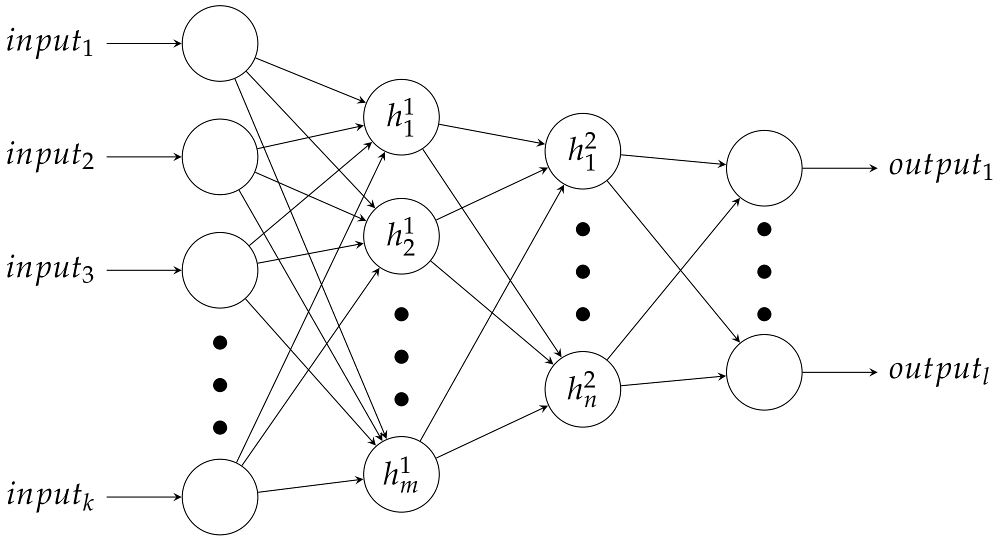


Figure 2.4: A feedforward neural network with k inputs and l outputs. There exist two hidden layers namely, h^1 and h^2, in which there exist m and n units respectively.

$$y = max(x, 0) \tag{2.16}$$

Feedforward neural network. After introducing the neuron, we introduce several fundamental structures for neural networks. One straightforward way is to cluster neurons into layers, and construct the network by fully connected neurons from neighboring layers, leading to one of the basic architecture known as the feedforward neural network, whose structure is summarized in Figure 2.4. More formally, different neurons are organized into layers, and each neuron is fully connected with all neurons in its adjacent layers, e.g., the connections among h^1 and h^2 in Figure 2.4. Weights associated with the connections represent the strengths of the connections, which are trained to encode patterns from training data.

Convolutional Neural Network (CNN). For some applications, it could be infeasible and unnecessary to learn a fully connected network as in feedforward network, given that one may not be interested in all connections in the first place. This is especially true for problems where the connections only exist among a small number of neurons, e.g., pixels in a small region from a image. Thus, the CNN architecture is proposed to cater for such scenarios. A CNN network looks similar to a feedforward network, except for the introduction of a spatially-local connection between neurons from adjacent layers. In other words, the neurons in layer h^m are only connected to neurons in a small contiguous region in layer h^{m-1}. Therefore, such kinds of architectures like CNN are also known as a locally-connected network. We refer readers to read (Goodfellow et al., 2016) for further details about the CNN architectures.

Recurrent Neural Network (RNN) is another way to organize neurons, specially catering for the sequential data. In feedforward networks there exist no feedback mechanism, namely, it can not express the sequential dependency in between adjacent states in the data. A typical example for such sequential data is the free text from a document, namely, the term at each offset depends on the one before (after) it. For example, a name of a city is more likely to appear after a preposition like "in". To model such dependencies, one would like to make a neuron only connected with some neurons representing the terms occur before or after it. One downside of RNN is the gradient vanishing and explosive problems, as a results of this recursive structure, making it hard to learn when the sequence becomes long. Different variants of RNN, including LSTM (Long Short Term Memory) (Hochreiter and Schmidhuber, 1997) and GRU (Gated Recurrent Unit) (Cho et al., 2014), have been proposed to especially tackle the training difficulties, by outweighing the long-term dependency with gate mechanisms. For details about the RNN and its variants, we refer readers to read the books and tutorials about RNN models (Goodfellow et al., 2016).

3 Low-cost Evaluation with Graded Judgments

3.1 Introduction

Manual labeling is laborious and costly, in particular when the number of topics and the number of compared systems are large. Thus, one desires to reduce the manual efforts in this procedure. The established measures, as introduced in Section 2.2.4, consume judgments as an abstraction of individual documents, and reward rankings with more relevant and diversified documents relative to the information need. Thus, given a ranking of documents for evaluation, the computation of different measures can be summarized as a sum of scores over judged documents in this ranking. However, on one hand, when few judgments are available, this computation tends to be inaccurate; on the other hand, the established measures can only employ the judged documents included in the evaluated document rankings. In other words, the established measures require all documents that have occurred in the evaluated rankings to be judged, and fail to utilize the judged documents that are not included in the rankings. Given the expensiveness of such manual judgments, one may desire to overcome such limitations and utilize manual judgments more effectively.

Recent years have seen a fair amount of research that seeks to reduce the cost of information retrieval evaluation. *Selective labeling*, as a first direction, chooses a subset of returned result documents to label. Among the simplest strategies, depth-k pooling only collects labels for documents returned in the top-k result of any of the compared systems. More sophisticated strategies leverage knowledge about the retrieval effectiveness, for instance, (Carterette et al., 2006) selectively labeled only documents with a potential effect on the relative order of any two systems. While cutting costs, selective labeling leads to result documents whose relevance label is not known. Such incomplete labels can also arise for other reasons, for example, when evaluating a novel information retrieval system that did not contribute to the original pool of result documents. *Mitigating incomplete labels*, as a second direction, seeks principled ways to make up for missing relevance assessments. The default of dealing with them is to assume that result documents are irrelevant if they have not been labeled. While this may appear pessimistic at first glance, it is not unreasonable given that most documents will be irrelevant to any specific information need. Alternative approaches have come up with novel effectiveness measures (Buckley and Voorhees, 2004), removed documents without known label from consideration (Sakai, 2007), and made use of machine learning methods to predict missing labels (Büttcher et al., 2007). Sakai et al. further explored the mitigation for novelty and diversity, investigating the reusability of test collections. Specifically, in (Sakai et al., 2012), the reusability was examined in terms of employing assessments collected with different pooling depths, and of system bias in a leave-one-out experiment as in (Büttcher et al., 2007). A condensed-list method from (Sakai, 2007) was employed to mitigate the missing judgments, by removing unjudged documents from the ranking before evaluation. It has been demonstrated that a condensed list can address this incomplete judgments issue when a significant amount of judgments, e.g., more than 50%, is still available.

Intuitively, similar to what is described in the *cluster hypothesis* (Jardine and Rijsbergen, 1971), documents that are relevant to the same query are supposed to be more similar with each other. Thus, as a *first contribution*, we investigate the agreement of documents in the pool with regard to the *cluster hypothesis* under different document representations, and better document representations are desirable to satisfy the aforementioned properties, ultimately in favor of low-cost evaluation. Beyond the representations, what has received some prior attention but has not been fully explored, though, is how the different strategies for selective labeling and incomplete label mitigation interact with each other.

As a *second contribution*, we examine the interaction of the state-of-the-art selective labeling and incomplete label mitigation strategies on four years of TREC Web Track data (2011–2014). The performance of different combinations is studied both in terms of approximating MAP scores (in terms of root mean square error) as well as system rankings (in terms of Kendall's τ). Also, strategies for selective labeling have typically been designed without consideration of how incomplete labels are dealt with later on. Furthermore, inspired by a recent work in machine learning (Yu et al., 2006) and the cluster hypothesis (Rijsbergen, 1979), we propose MAXREP as a novel selective labeling strategy. MAXREP selects documents to label so as to maximize their representativeness of the pool of result documents, thus yielding effective training data for label prediction. MAXREP is formulated as an optimization problem, which permits efficient approximation.

As a *third contribution* following the work in (Sakai et al., 2012), we propose a novel measure framework, named LMD-CASCADE, which directly utilizes the language model of all available judged documents. LMD-CASCADE approximates established cascade measures for novelty and diversity but is robust when faced with incomplete judgments or another document collection. Specifically, instead of representing the documents by their manual judgments and encoding the procedure with formula in a measure, we directly analogize this cascade procedure with the relationships between the language models based on a subtopic and on the search results. In total, we end up with four novel cascade measures coined ABSNB, ABSRB, DELTANB, and

DeltaRb. As a difference to existing measures which explicitly penalize redundancy in the form of repetitions of the same subtopic, our measures implicitly capture redundancy via the estimated language models. This part complements the established works for incomplete judgments in the context of effectiveness measures for novelty and diversity.

3.2 Related Work

Several efforts have looked into how, to reduce human effort and hence cost, only a subset of returned documents can be labeled, while still producing a reliable relative ranking of multiple information retrieval systems.

3.2.1 Document Representations

In this section, we recap different established document representations described in Section 2.3.1.

- **Bag-of-words sparse vector with tf-idf weighting** (Bow) (Büttcher et al., 2007). Each document is represented as a sparse word vector, with components determined by tf-idf weighting.
- **Latent semantic analysis** (Lsa) (Landauer et al., 1998). We decompose the document-term matrix and employ the dense document vectors.
- **Latent Dirichlet Allocation** (Lda) (Blei et al., 2003). A generative model to capture co-occurrences of terms among different documents. The GibbsLDA++ toolkit is employed in our experiments (Phan and Nguyen, 2007).
- **Neural network based document vectorization** (Para2vec) (Le and Mikolov, 2014). The recently proposed Para2vec method co-trains the document vector together with the word vectors. The Para2vec is trained with gensim toolkit (Řehůřek and Sojka, 2010) in this work.

3.2.2 Selective Labeling

As a second part, we briefly review different selective labeling methods. *Pooling strategies* merge the results returned by different systems to form a pool of result documents to be labeled by human assessors. The most common strategy, *depth-k pooling* as used by TREC, considers only documents that are returned within the top-*k* of any system. Cormack et al., as an alternative, proposed *move-to-front pooling* (Cormack et al., 1998) as an iterative pooling procedure, requiring continued human effort, which systematically prioritizes documents returned by systems that have already returned relevant documents. Vu and Gallinari made use of machine learning for pooling (Vu and Gallinari, 2006). Using documents from the top-5 pool as training data, they employed *learning-to-rank methods* to estimate the relevance of yet-unlabeled documents. Documents more likely to be relevant are then labeled with higher priority. Features, in their case, encode the rank at which the document was returned by different systems. Their approach thus requires two rounds of human interaction to label (i) documents in the top-5 pool as training data and (ii) a number of the remaining documents. Aslam et al. devised a biased sampling strategy that yields an unbiased estimator of MAP (Aslam et al., 2006). A more practical sampling strategy with good empirical performance is described in (Pavlu and Aslam, 2007). The key idea here is to introduce a sampling distribution, so that documents ranked highly by many system, which are therefore expected to be relevant, are selected more often. The probability of selecting the document at rank r from a result list of length n is defined as

$$P[r] \approx \frac{1}{2n} \log \frac{n}{r} \, .$$

These per-system probabilities are aggregated, corresponding to choosing a system at uniform random, and documents are selected using stratified sampling. Carterette et al. proposed the *minimal test collection* (MTC) method (Carterette et al., 2006). For a specific retrieval effectiveness measure (e.g., MAP or nDCG), MTC iteratively selects discriminative documents to label until the relative order of systems has been determined. Requiring continued human interaction at every step, like move-to-front pooling described above, it is an active procedure.

Unlike all of the aforementioned strategies, which only take ranking information into account, the proposed method MaxRep also considers document contents. Inspired by (Yu et al., 2006) and designed with label prediction in mind, MaxRep aims at selecting a representative set of documents from the pool of result documents to yield effective training data.

3.2.3 Incomplete Label Mitigation

Labels can be incomplete for different reasons, for instance, since they were collected only selectively or because the evaluated information retrieval system is novel and did not contribute to the initial result pool. Different strategies have been proposed as remedies.

As already mentioned, a common way to deal with missing relevance labels, which is also used in TREC, is to *assume* that those documents are *irrelevant*. Given that most documents are irrelevant anyway for any specific information need, this can also be interpreted as label prediction with a simple majority classifier. More elaborate label prediction methods will be discussed below. Sakai, as an alternative, proposed to remove documents without known labels from consideration yielding *condensed result lists* (Sakai, 2007). Both aforementioned incomplete label mitigation strategies are agnostic to the retrieval effectiveness measure used. In contrast, Buckley and Voorhees proposed bpref as an *alternative retrieval effectiveness measure* mimicking mean-average precision (MAP) (Buckley and Voorhees, 2004). With R as the number of labeled relevant documents, it is defined as

$$\text{bpref} = \frac{1}{R} \sum_r \left(1 - \frac{|\,\text{labeled irrelevant above rank } r\,|}{R}\right) ,$$

and the term in parenthesis can be interpreted as an estimator of precision at rank r. In their experiments, bpref proved robust and exhibited high rank correlation with MAP. However, in terms of numerical value, bpref may deviate from MAP if many labels are missing. Yilmaz and Aslam described two alternatives, based on sampling theory, that are closer to MAP (Yilmaz and Aslam, 2006). The first, induced average precision (indAP), removes documents with unknown label from consideration and can be seen as an application of the condensed list approach (Sakai, 2007) to MAP. The second, inferred average precision (infAP), relies on the following improved estimator of precision at rank r

$$E[\text{precision at } r] = \frac{1}{r} + \frac{(r-1)}{r} \left(\frac{|\,\text{labeled above rank } r\,|}{r-1} \cdot \frac{|\,\text{labeled relevant}\,|}{|\,\text{labeled}\,|} \right)$$

which also takes into account what fraction of documents has been labeled.

Another family of strategies uses machine learning methods to *predict missing relevance labels*. Carterette and Allan used regularized logistic regression to predict the relevance of documents (Carterette and Allan, 2007). Building on the cluster hypothesis (Rijsbergen, 1979), document features encode *tf-idf* based cosine similarity with documents whose labels are known. Büttcher et al., to the same end, explored two approaches, namely a simple classifier based on statistical language models and a support vector machine (SVM) (Büttcher et al., 2007). For the latter, document features are *tf-idf* weights for the 10^6 most common terms in the document collection. Given the good performance of the SVM-based label prediction in their experiments, we use this as one of the incomplete label mitigation strategies in our experiments. In addition, Sakai et al. further considered the incomplete judgments particular for novelty and diversification (Sakai et al., 2012). where the situations of incomplete judgments from leave-one-out experiments and from expansion of judgment pooling are examined. In both cases, however, available judgments are beyond 50% of total judgments. The works in (Sakai et al., 2012) and (Sakai, 2013) are closest to the proposed LMD-CASCADE in the sense that the incomplete judgments over diversification is considered, and it can be regarded as a complement to (Sakai et al., 2012) when established cascade measures fail to work.

3.3 Dataset

Our experiments are based on the ClueWeb09[1] and ClueWeb12[2] document collections, which include 500 and 700 million English web pages respectively. Queries and relevance labels are taken from the adhoc and diversity tasks of the TREC Web Track (2009–2014). This leaves us with a total of 300 queries (50 per year) and their corresponding relevance labels. In total, there are 300 queries and 113k judgments (qrels). We also obtained the runs submitted by participants of the TREC Web Track. There are 71 runs for 2009, 55 runs for 2010, 62 runs for 2011, 48 runs for 2012, 61 runs for 2013, and 42 runs for 2014. For each submitted run we consider the top-20 search results returned.

On ClueWeb09, the complete set of 500 million English web pages is known as ClueWeb09 Category A (CwA). For our robustness and reusability experiments in Section 3.6.2, we also make use of ClueWeb09 Category B (CwB), as a well-defined subset of about 50 million English web pages. As a third subset of English web pages, coined CwC, we consider all 450 million web pages that are part of CwA but not CwB.

3.4 Cluster Hypothesis versus Document Representations

3.4.1 Method

As mentioned, one way to reduce the manual effort is to introduce a semi-automatic method for labeling documents with regard to their relevance to a given query (Carterette and Allan, 2007). Given a document representation, the following properties are desirable for document similarity: 1) Given a query, the relevant documents should be more similar with each other than with the non-relevant documents; 2) Further, for ambiguous or multi-faceted queries, the documents that are relevant to the same subtopic(s) should be more similar with each other. Note that, the boundary between different types of documents is emphasized in the aforementioned properties. In reality, though, our pilot experiments indicate that the inter-document similarity is far from perfect for low-cost evaluation. One crucial reason for that is the loss of information in representing documents. No matter which low-cost evaluation methods are used, documents need to be firstly represented for all kinds of follow-up computations. For example, the bag-of-words representation with tf-idf weighting is widely used, but its assumption about the independence among terms leads to sparsity issues.

[1]http://www.lemurproject.org/clueweb09.php/
[2]http://www.lemurproject.org/clueweb12.php/

Task	Benchmark	Bow	Ebow	Lda	Lsa	Para2vec
Adhoc Task	TripleTest	0.61	**0.62 (1.1%)**	0.51 (-17%)	0.47 (-24%)	0.53 (-14%)
	KnnTest: $k = 5$	0.62	0.62 (0.4%)	0.53 (-15%)	0.58 (-6%)	0.57 (-8%)
	KnnTest: $k = 20$	0.54	0.54 (0.6%)	0.44 (-18%)	0.47 (-12%)	0.46 (-15%)
Diversity Task	TripleTest	0.49	**0.51 (4.5%)**	0.43 (-13%)	0.44 (-10%)	**0.51 (4.1%)**
	KnnTest: $k = 5$	0.65	0.65 (-0.1%)	0.58 (-11%)	0.64 (-0.8%)	0.63 (-2.9%)
	KnnTest: $k = 20$	0.56	0.56 (-0.1%)	0.51 (-8%)	0.54 (-3.5%)	0.54 (-4.5%)

Table 3.1: Comparison of Document Representations on Different Benchmarks

In this section, we investigate the agreement of documents relative to the *cluster hypothesis*. We compare multiple document representations, including bag-of-words, latent semantic analysis (Landauer et al., 1998), latent dirichlet allocation (Blei et al., 2003) and the recently proposed Para2vec (Le and Mikolov, 2014) methods on different benchmarks. In addition, inspired by the recent success of neural network based word embedding method (Mikolov et al., 2013) in capturing semantic similarity among terms, we try to utilize the term embedding in representing documents, transferring the powerful term similarity to the document level to mitigate the issue of sparsity. In particular, we denote the proposed representation as **the bag-of-word sparse vector expanded with similarity among term embeddings** (Ebow). Therefore, to mitigate the sparsity of Bow, we encode the term embeddings from Word2vec (Mikolov et al., 2013) by expanding the Bow with similarity among term vectors.

3.4.2 Results

In the adhoc task, all 200 queries and all documents from *qrel* are used. In the diversity task, 145 queries annotated with more than one subtopic and documents that are relevant to at least one subtopic are used. In Lsa, Lda and Para2vec, the document representation is computed separately for each query, given the size of the complete ClueWeb dataset. The results summarized in Table 3.1 are the average results among queries, with **bold numbers** indicating statistically significant improvements when compared against Bow. Intuitively, the comparisons among different document representations are in terms of their agreement degree to the desirable properties mentioned in the introduction, where cosine similarity is used. To measure this agreement, the following benchmarks are employed. **Direct comparison of similarity value (TripleTest).** To employ the document similarity in low-cost evaluation, the most important part is to distinguish document pairs that are both relevant to the query and those including one relevant and one non-relevant document. Thereby, we follow the document similarity benchmark used in (Le and Mikolov, 2014). In particular, for each query q, document triples (d_{r1},d_{r2}, d_n) are created from *qrel*, such that d_{r1} and d_{r2} are relevant to q, or relevant to the same subtopic(s), and d_n is

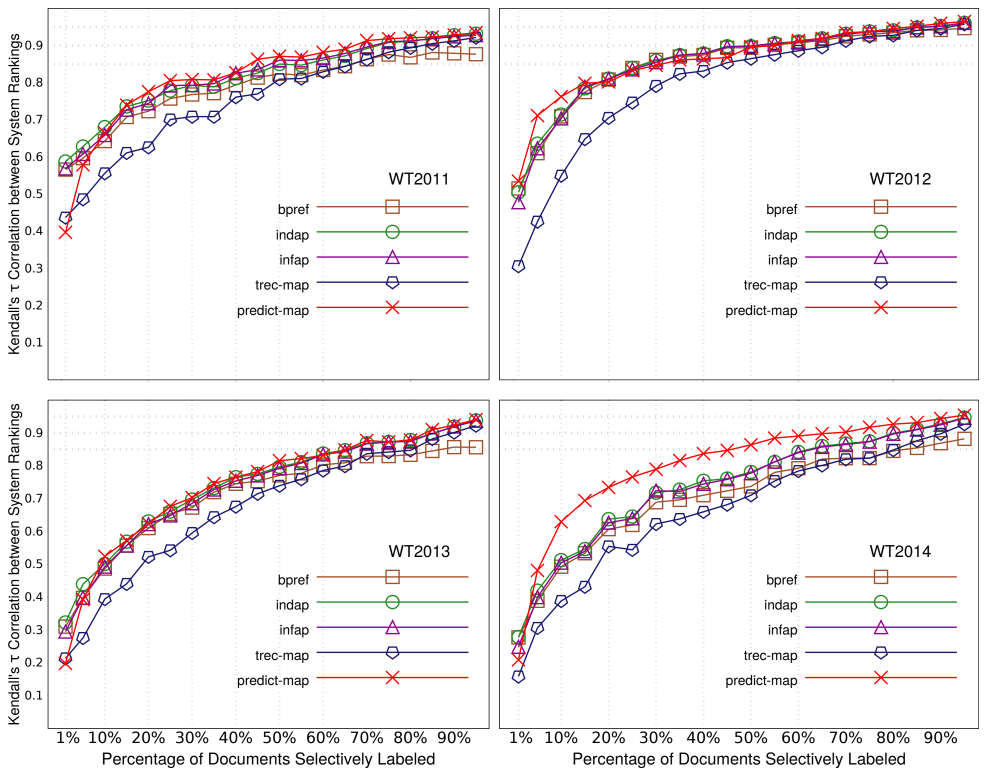


Figure 3.1: Approximation of system rankings with **uniform random sampling**: X-axes indicate percentages of labeled documents. Y-axes indicate Kendall's τ correlation.

non-relevant, or is relevant to different subtopics from d_{r1} and d_{r2}. Similar to the metric used in (Le and Mikolov, 2014), if d_{r1} and d_{r2} are more similar with each other than with d_n, the document triple is regarded correct. Different methods are compared based on the aggregated ratio between the correct triples and the total triples among queries. **Near-neighbor test (KnnTest).** Introduced in (Voorhees, 1985), the ratio of relevant documents among the k closest neighbors for each relevant document are examined. In this section, we examine this relevant document ratio for different $k \in \{5, 20\}$.

Table 3.1 shows that the agreement is not good enough in terms of absolute value, e.g., on TripleTest, 0.6 indicates that the boundary of relevant and non-relevant documents is blurred, and better representations are desirable to fulfill the low-cost evaluation task. Moreover, the results also indicate that improving document representations with word embeddings is non-trivial: Ebow improve Bow on TripleTest by **1.1%** and **4.5%** respectively, meanwhile Para2vec (Le and Mikolov, 2014) performs worse on adhoc task.

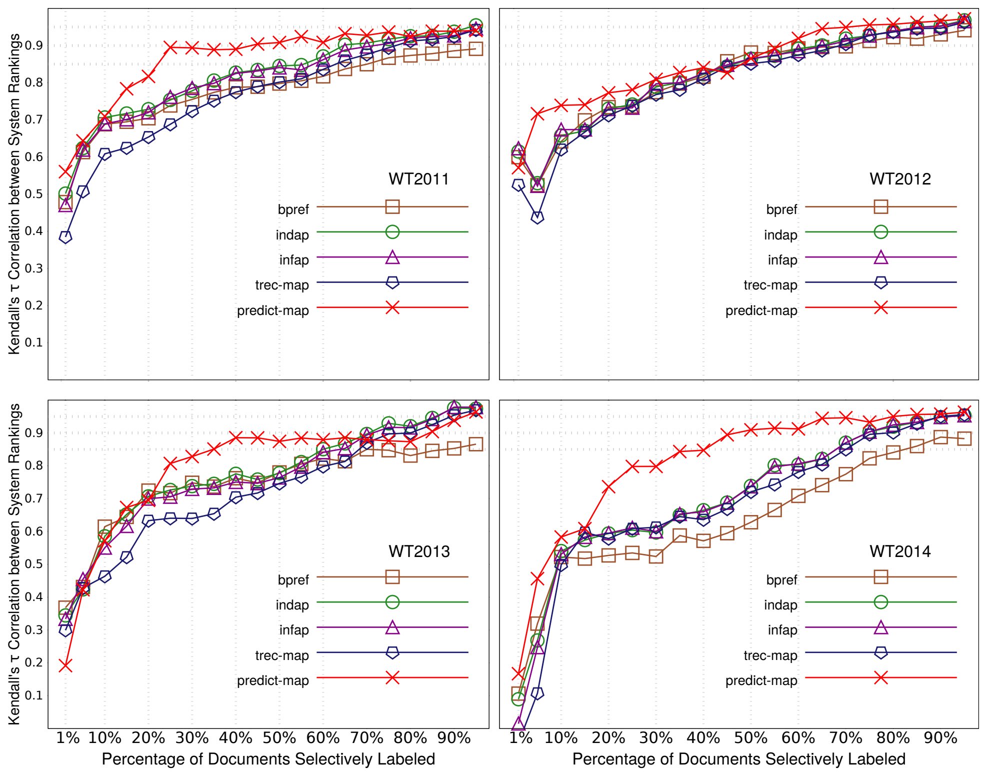


Figure 3.2: Approximation of system rankings with **incremental pooling**: X-axes indicate percentages of labeled documents. Y-axes indicate Kendall's τ correlation.

3.5 MaxRep: A Selective Labeling Method for Ad-hoc IR

3.5.1 Method

In this section, we describe MaxRep, our novel strategy for selective labeling. In contrast to existing strategies, MaxRep not only considers ranking information but also takes into account document contents. Intuitively, it aims at selecting a subset of documents that is representative, in particular of those documents expected to be relevant. MaxRep thus harvests effective training data for label prediction, since documents are representative of the overall pool of result documents, and it also makes up for the inherent bias against relevant documents.

Let $\mathcal{D}$ denotes the pool of result documents for a specific topic. Our objective is to select a k-subset $\mathcal{L} \subseteq \mathcal{D}$ that best represents the pool of result documents. Intuitively, if two documents have similar contents, there is no need to label both of them, since their labels tend to be identical. We let $sim(d_i, d_j) \in [0, 1]$ denote a measure of *content similarity* between documents d_i and d_j. Further, we let $rel(d_i) \in [0, 1]$ denote a measure of *expected relevance* of document d_i

Our concrete implementation uses the cosine similarity between $tf - idf$ document vectors as a

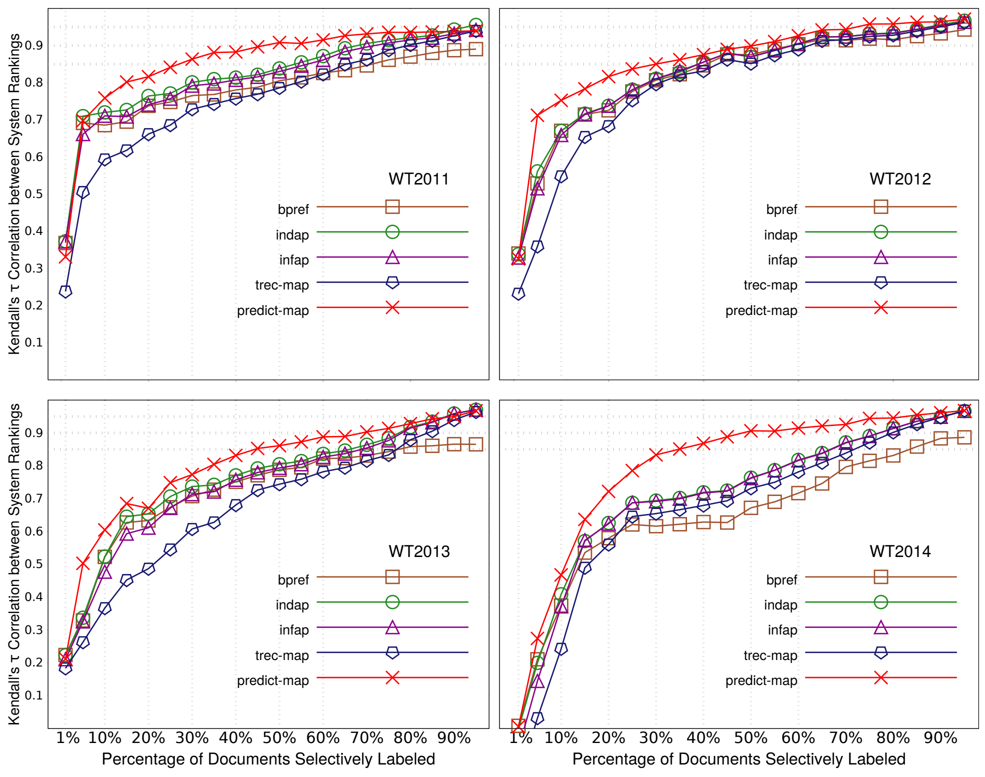


Figure 3.3: Approximation of system rankings with **statAP**: X-axes indicate percentages of labeled documents. Y-axes indicate Kendall's τ correlation.

measure of document content similarity, as defined in Equation 2.1. Thus, the similarity between documents d_i and d_j is computed as in Equation 3.1.

$$sim(d_i, d_j) = \frac{\vec{d_i} \cdot \vec{d_j}}{\|\vec{d_i}\| \, \|\vec{d_j}\|} \tag{3.1}$$

which ranges in $[0, 1]$ given that we only have non-negative feature weights. As in (Büttcher et al., 2007) our implementation only considers the 10^6 most frequent terms from the document collection. Moreover, in order to reduce noise, we ignore similarities below 0.8, setting them to zero, when choosing representative documents. As a measure of expected relevance our concrete implementation uses the probability according to the sampling distribution also used in (Pavlu and Aslam, 2007) and described in Section 3.2.2. We measure the representativeness of a document set $\mathcal{L}$ as

$$f(\mathcal{L}) = \sum_{d_i \in \mathcal{D}} rel(d_i) \max_{d_j \in \mathcal{L}} \left(sim(d_i, d_j) \right) \ . \tag{3.2}$$

This formulation rewards document sets that cover all documents from $\mathcal{D}$ that are expected to be relevant by including at least one similar document.

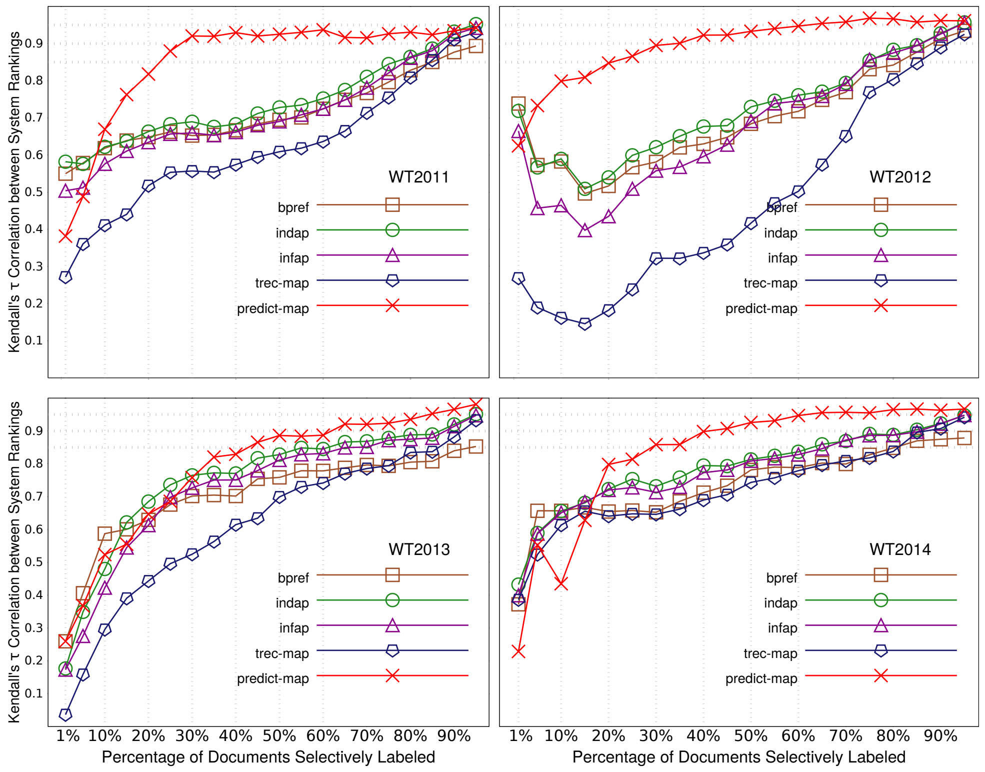


Figure 3.4: Approximation of system rankings with **MaxRep**: X-axes indicate percentages of labeled documents. Y-axes indicate Kendall's τ correlation.

Building on this, we cast selecting the set of k most representative result documents into the optimization problem described in Equation 3.3. It turns out that the above optimization problem permits efficient approximation thanks to the submodularity of its objective function, which we state in the Lemma 3.5.1. Having established the submodularity of our objective function in the proof of Lemma 3.5.1, we can make use of the result in (Nemhauser et al., 1978) and greedily build up the set of representative documents $\mathcal{L}$. More precisely, starting from $\mathcal{L}_0 = \emptyset$, in the i-th iteration we include the document from $\mathcal{D} \setminus \mathcal{L}_{i-1}$ that maximizes $f(\mathcal{L}_i)$, and finally report $\mathcal{L}_k$ as our result. This greedy algorithm gives a $(1 - \frac{1}{e})$-approximation guarantee.

$$\underset{\mathcal{L}}{\operatorname{argmax}}\, f(\mathcal{L}) \quad \text{s.t.} \quad |\mathcal{L}| = k \tag{3.3}$$

Lemma 3.5.1 (Submodularity)**.** *Equation 3.2 defines a submodular function. Given two document sets $\mathcal{L}$ and $\mathcal{L}'$ with $\mathcal{L} \subseteq \mathcal{L}'$ and a document $d \in \mathcal{D}$, then*

$$f(\mathcal{L} \cup \{d\}) - f(\mathcal{L}) \geq f(\mathcal{L}' \cup \{d\}) - f(\mathcal{L}') \, .$$

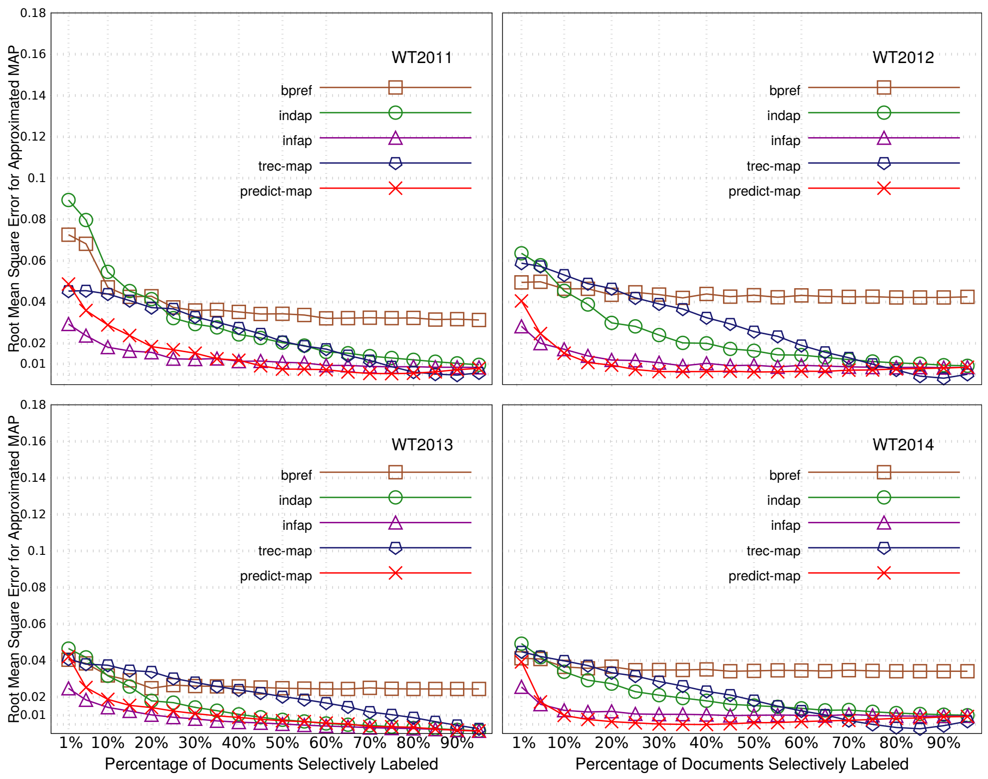


Figure 3.5: Approximation of MAP scores with **uniform random sampling**: X-axes indicate percentages of labeled documents. Y-axes indicate root mean square error (RMSE).

Proof of Lemma 3.5.1. We can rewrite for $\mathcal{X} \in \{\mathcal{L}, \mathcal{L}'\}$

$$f(\mathcal{X} \cup \{d\}) - f(\mathcal{X}) = \sum_{d_i \in \mathcal{D}} rel(d_i) \max \left(0, sim(d_i, d) - \max_{d_j \in \mathcal{X}} sim(d_i, d_j)\right) .$$

Now,

$$\begin{aligned} \mathcal{L} \subseteq \mathcal{L}' \quad &\Rightarrow \quad \forall d_i \in \mathcal{D} : \max_{d_j \in \mathcal{L}} sim(d_i, d_j) \leq \max_{d_j \in \mathcal{L}'} sim(d_i, d_j) \\ &\Rightarrow \quad f(\mathcal{L} \cup \{d\}) - f(\mathcal{L}) \geq f(\mathcal{L}' \cup \{d\}) - f(\mathcal{L}') . \end{aligned}$$

□

3.5.2 Result

In this section, we describe our experimental evaluation. We report on the performance of different combinations of strategies for selective labeling, including MaxRep as the one proposed in Section 3.5.1, and incomplete label mitigation. This is done on four years' worth of participant data from the TREC Web Track (2011–2014), and we investigate how well combinations can

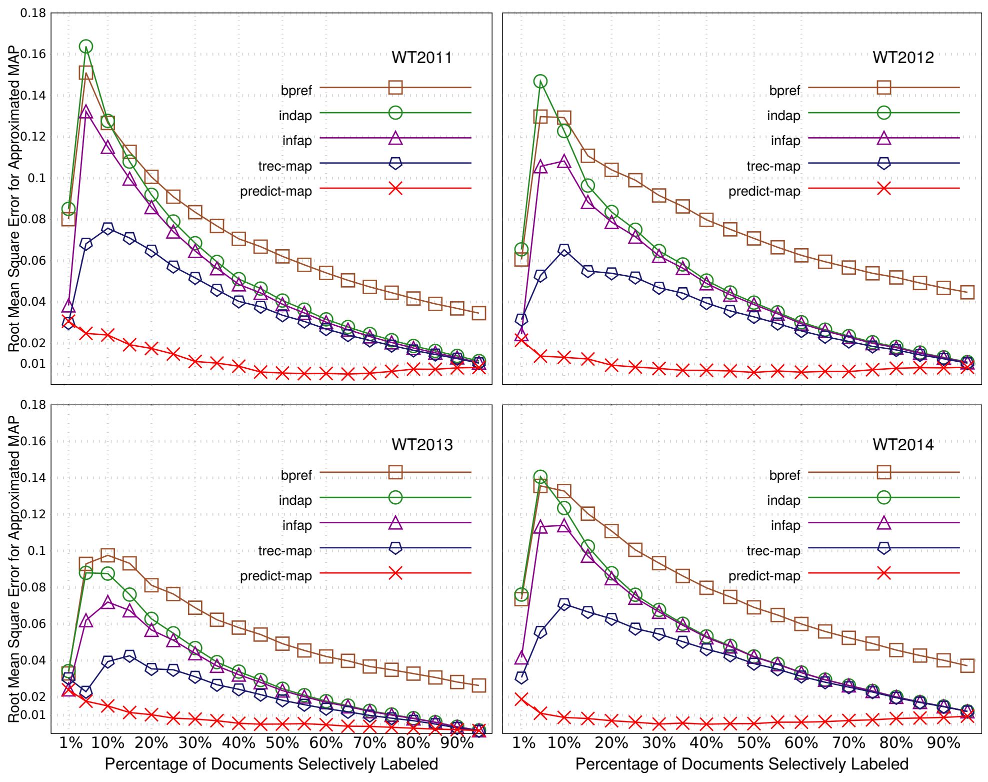


Figure 3.6: Approximation of MAP scores with **incremental pooling**: X-axes indicate percentages of labeled documents. Y-axes indicate root mean square error (RMSE).

approximate the system ranking, in terms of Kendall's τ, but also how well they can approximate MAP scores, in terms of root mean square error (RMSE).

Competing Methods

We consider the following non-active strategies for *selective labeling*:

- **uniform random sampling**, as described in (Buckley and Voorhees, 2004), we give the method an advantage by sampling retrospectively from relevant and irrelevant documents (we report averages based on 30 repetitions);
- **incremental pooling**, as described in (Carterette et al., 2006) and (Carterette, 2007), we select documents according to the best rank assigned by any system and break ties according to the average rank across all systems;
- **statAP**, as described in (Pavlu and Aslam, 2007), with additional judgments obtained from pooling (we report averages based on 30 repetitions);
- **our** MAXREP as described in Section 3.5.1.

To *mitigate incomplete labels*, we consider the following strategies:

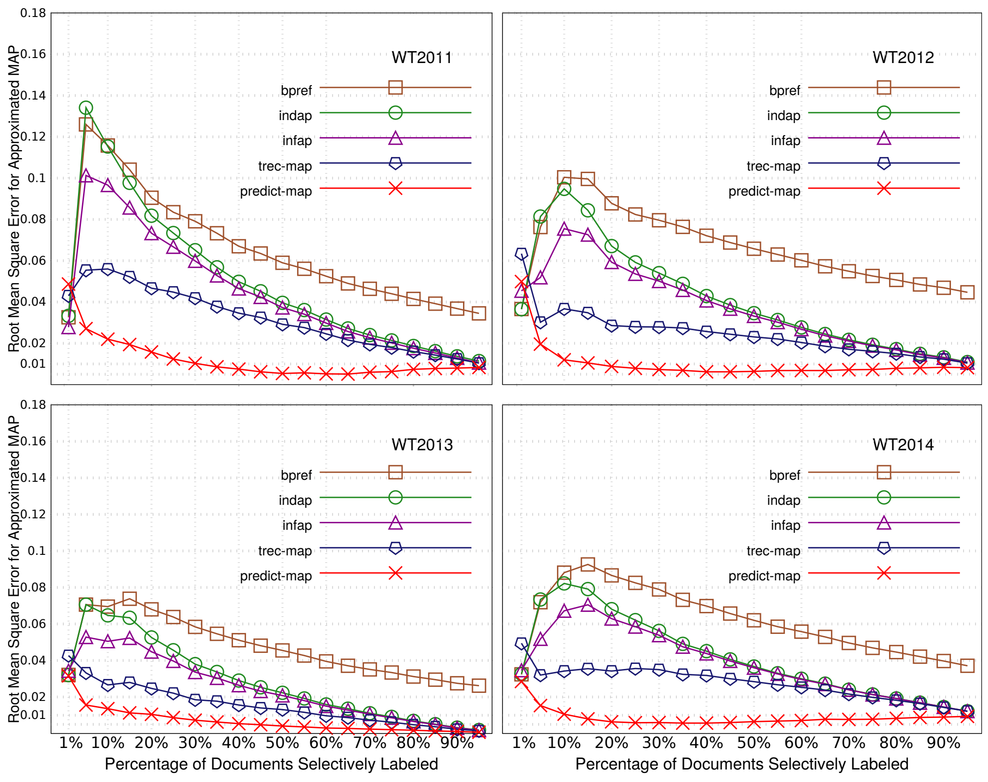


Figure 3.7: Approximation of MAP scores with **statAP**: X-axes indicate percentages of labeled documents. Y-axes indicate root mean square error (RMSE).

- **trec-map** directly regards unlabeled documents as non-relevant, which is also the default setting in TREC;
- **bpref** (Buckley and Voorhees, 2004) further separates the labeled non-relevant documents from unlabeled documents, considering the effects of the labeled non-relevant documents;
- **indAP** (Yilmaz and Aslam, 2006) regards missing labels as non-existing, by only considering the labeled documents in computing the average precision;
- **infAP** (Yilmaz and Aslam, 2006) estimates the average precision by computing the expectation using sampled relevance judgments;
- **statAP** (Pavlu and Aslam, 2007) computes AP with adjustments by including probability from the document sampling phase;
- **predict-map** predicts the unknown labels with SVM-based label prediction approach as in (Büttcher et al., 2007), which we implemented using the scikit-learn (Pedregosa et al., 2011) toolkit.

This gives us a total of 21 combinations to investigate. Given that statAP as a strategy for mitigating incomplete labels requires inclusion probabilities as an input from selective labeling, we only compute statAP when labels have been selected with statAP itself.

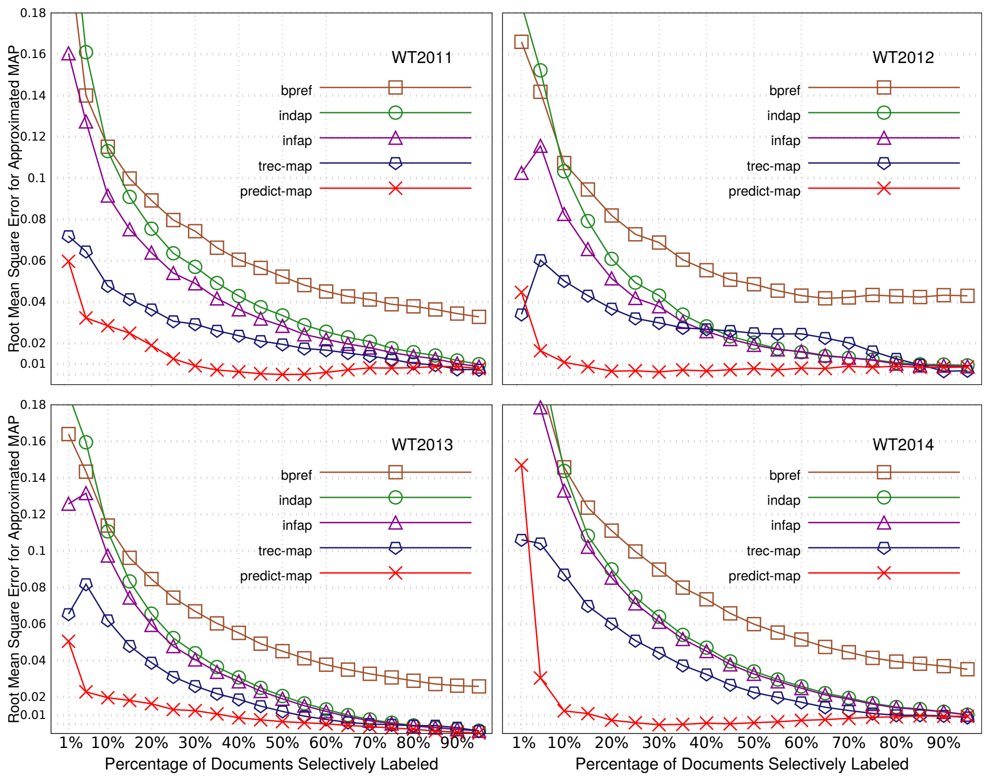


Figure 3.8: Approximation of MAP scores with **MaxRep**: X-axes indicate percentages of labeled documents. YY-axes indicate root mean square error (RMSE).

Approximation of System Ranking and MAP Scores

Our first experiment studies how well different strategies can approximate the system ranking in terms of Kendall's τ and how well they can approximate the MAP scores of individual systems. We select a varying percentage, from 1% up to 95%, to label using the different strategies. Figures 3.1, 3.2, 3.3 and 3.4 show the Kendall's τ values obtained for different selective labeling strategies on each of the four years (2011–2014) considered. From these figures, it can be seen that the right combination between the selective labeling methods and the computation of evaluation measures are crucial and there is even no guarantee that a more reliable evaluation could be reached when more judgments are available. This is especially true when very few judgments are available, making the evaluation a bold guess. For example, in Figure 3.4, when MAXREP is employed for selective labeling, the reliability of original MAP (**trec-map**) becomes really poor, e.g., in WT2012. This is because the documents are selectively labeled according to their "prediction power" without considering the reliability and the robustness of the MAP based on such judgments. In an extreme case, assuming all selectively labeled documents exclusively come from one single run, the corresponding MAP score for other runs would become totally meaningless.

Comparing the different incomplete label mitigation strategies, we observe that predict-map, the

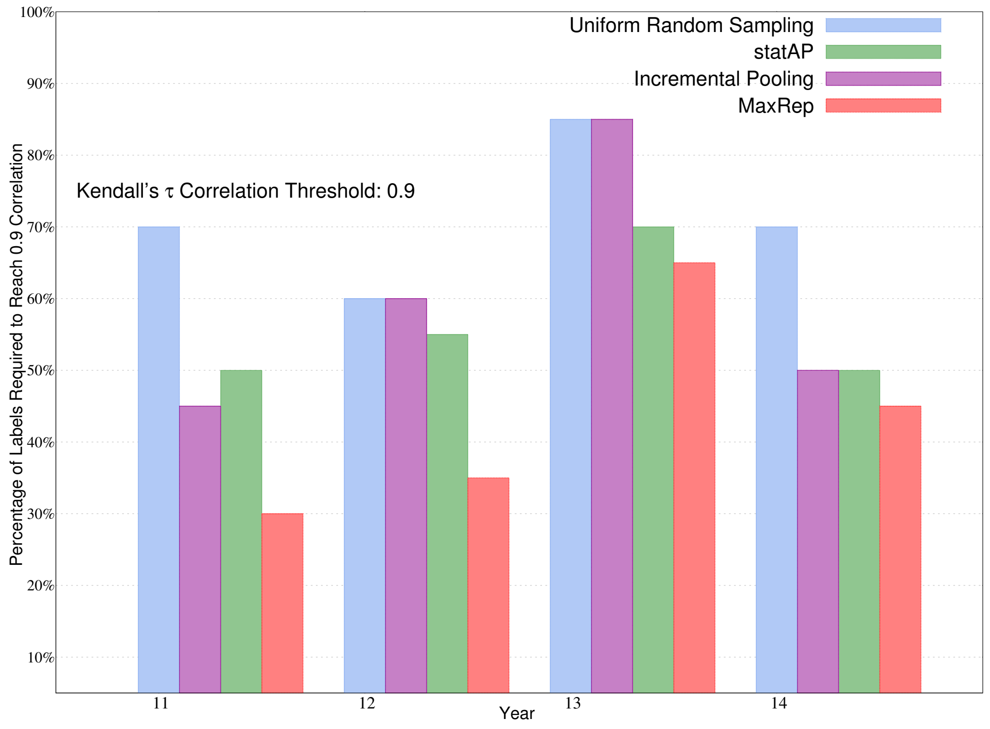


Figure 3.9: Percentage of labeled documents required to achieve a Kendall's τ correlation above 0.9 when using label prediction.

SVM-based label prediction approach, consistently achieves good performance, regardless of how documents to label are selected. In most plots, with as little as 20% of labeled documents, predict-map thus achieves a Kendall's τ value above 0.9, which indicates that the obtained system ranking is practically indistinguishable from the ground truth. Using trec-map and assuming that documents without known labels are non-relevant, totally mixing the labeled non-relevant and unlabeled documents, at the other extreme, performs worst in most cases. Not surprisingly, this is most pronounced when using our selective labeling strategy MAXREP. Figures 3.5, 3.6, 3.7 and 3.8 plot the corresponding root mean square error (RMSE), measuring how well the different combinations approximate MAP scores of individual systems. Predicting missing labels using predict-map again achieves the best result by yielding the lowest approximation errors. The highest approximation errors are almost consistently seen for bpref, which is not surprising given that, as described in Section 3.2.3, it is different from MAP.

Selective Labeling under Label Prediction

Given the good performance of label prediction in the previous experiment, we now investigate which selective labeling strategy performs best with it. In Figure 3.9, we plot the percentages of documents that need to be labeled, with different selective labeling strategies, when using predict-map for label prediction to achieve a Kendall's τ score above 0.9.

As can be seen, our selective labeling strategy MaxRep performs best across all four years under consideration. It thus consistently requires the lowest percentage of documents to be labeled to achieve a system ranking that is practically indistinguishable from the ground truth. Its relative advantage is clearest for the years 2011 and 2012 for which MaxRep requires as little as $30 - 35\%$ of labeled documents. Also in this experiment, uniform random sampling performs worst, typically requiring more than 60% of labeled documents to achieve a Kendall's τ value above the threshold. Additionally, we conduct paired two-tailed t-tests between different baselines w.r.t. our method for these least percentage of labels required to get over 0.9 correlation, and our method outperforms the uniform random sampling and incremental pooling at 95% significant level (p-value=.008 and .032), meanwhile outperforms the statAP at 90% level (p-value=.063).

As for comparison on RMSE, from Figures 3.5, 3.6, 3.7 and 3.8, we can see that our method is comparable to other methods in terms of approximating MAP scores. However, no clear winner is observed among different selective labeling methods when combined with mitigation through label prediction.

3.6 Lmd-Cascade: A Cascade Measure Framework

3.6.1 Method

Having described existing cascade measures for novelty and diversity in Section 2.2.4, we introduce a family of novel measures in this section. In contrast to the existing ones, which directly digest relevance judgments provided by humans, our measures operate on statistical language models estimated for each subtopic based on its relevant documents as well as the top-k result documents. Instead of directly trying to predict missing relevance judgments, as the **predict-map** from Section 3.5.2, we directly derive gain values from the Kullback-Leibler divergences between those language models estimated and aggregate them taking positions of result documents into account. This indirect approach makes our measures robust and capable of dealing with substantially incomplete relevance judgments, as we will demonstrate in our experimental evaluation.

Let us now introduce our formal notation. The document collection is denoted as $\mathcal{D}$ with documents therein as bags of words drawn from a vocabulary $\mathcal{V}$ of indexed terms. For a term $t \in \mathcal{V}$ we use $tf(t, d)$ to denote its term frequency in document $d \in \mathcal{D}$ and let $|d| = \sum_{t \in \mathcal{V}} tf(t, d)$ denote the document length. We refer to the subtopics of a query q as $\left\{ q_1, \ldots, q_{|q|} \right\}$ and let $r(q_i, d)$ be a predicate indicating the (binary) relevance of document d to subtopic q_i. Finally, we refer to a query result as $R = \langle d_1, \ldots, d_{|R|} \rangle$ and as $R_k = \langle d_1, \ldots, d_k \rangle$ to its corresponding top-k result.

Statistical language models have ample applications in natural language processing including tasks such as speech recognition and machine translation. More recently, within the last two decades, they have also been successfully applied for tasks in Information Retrieval – an overview of the state of the art is given in (Zhai, 2008b).

In this work, language models serve two purposes. First, they characterize what makes up a relevant document for a specific subtopic. Second, they capture what users see while sifting through the query result. While more advanced language models have been proposed (e.g., based on n-grams or allowing for term translations), for simplicity, we restrict ourselves to unigram language models with Dirichlet smoothing.

Top-k Query Result Language Model. From the top-k query result R_k we estimate a language model Θ_{R_k} as

$$P[t|\Theta_{R_k}] = \frac{\sum_{d \in R_k} tf(t,d) + \mu}{\sum_{d \in R_k} |d| + \mu\, P[t|\Theta_{\mathcal{D}}]} \,. \tag{3.4}$$

Here, μ is a tunable parameter (set as $\mu = 2{,}500$ in our experiments (Zhai, 2008b)) which controls the influence of Dirichlet smoothing with the language model $\Theta_{\mathcal{D}}$ estimated from the document collection as

$$P[t|\Theta_{\mathcal{D}}] = \frac{\sum_{d \in \mathcal{D}} tf(t,d)}{\sum_{d \in \mathcal{D}} |d|} \,. \tag{3.5}$$

The language model Θ_{R_k} thus captures what users see when eagerly inspecting all documents up to rank k. The smoothing with the document collection language model $\Theta_{\mathcal{D}}$ can be interpreted as their prior knowledge about general documents from the collection. By its definition, Θ_{R_k} captures the degree of diversity in the top-k query result. Intuitively, when homogeneous documents related to a single subtopic are returned, the estimated language model Θ_{R_k} will have lower entropy than in the case when heterogeneous documents related to various subtopic are returned. Moreover, Θ_{R_k} comes with an inherent bias against documents returned at lower ranks. When comparing Θ_{R_k} and $\Theta_{R_{k+1}}$ it is clear from the definition that the influence of the additional result document on the estimate decreases as k increases.

Subtopic Language Models. Likewise, given a query q and its subtopics $\left\{q_1, \ldots, q_{|q|}\right\}$, we estimate a language model

$$P[t|\Theta_{q_i}] = \frac{\sum_{d \in \mathcal{D}\,:\,r(q_i,d)} tf(t,d) + \mu}{\sum_{d \in \mathcal{D}\,:\,r(q_i,d)} |d| + \mu\, P[t|\Theta_{\mathcal{D}}]} \tag{3.6}$$

for each subtopic based on its known relevant documents, again smoothed with the document collection language model $\Theta_{\mathcal{D}}$. The purpose of smoothing is twofold, namely to avoid zero probabilities and to achieve a relative weighting of terms for the following divergence computation.

Divergence-based gain. We obtain gain values by comparing the language models estimated for subtopics and top-k results. To make this more precise, let Θ_{q_i} be a subtopic language model and Θ_{R_k} be a top-k query result language model estimated as described above. We use their Kullback-Leibler divergence

$$KLD(\Theta_{q_i} \parallel \Theta_{R_k}) = -\sum_{t \in \mathcal{V}} P[t|\Theta_{q_i}] \log \left(\frac{P[t|\Theta_{q_i}]}{P[t|\Theta_{R_k}]} \right) , \tag{3.7}$$

as a building block to compare language models. We thus obtain high values $KLD(\Theta_{q_i} \parallel \Theta_{R_k})$ when the top-k result documents in R_k are different from the relevant documents known for subtopic q_i, for instance, since they use different terminology or other key terms. To obtain a per-subtopic gain values, we transform the Kullback-Leibler divergences as

$$g(i,k) = max(0, 1 - \frac{KLD(\Theta_{q_i} \parallel \Theta_{R_k})}{KLD(\Theta_{q_i} \parallel \Theta_{\mathcal{D}})}) , \tag{3.8}$$

normalizing with the Kullback-Leibler divergence observed for the document collection language model $\Theta_{\mathcal{D}}$. In practice, the above per-subtopic gain value ranges in $[0, 1]$.

We consider two alternative formulations to turn these per-subtopic gain values $g(i,k)$ into per-rank gain values, which can then be aggregated. Our first formulation, coined ABS, determines a per-rank gain value as

$$g(j) = \max_{1 \leq i \leq |q|} g(i,j) , \tag{3.9}$$

thus rewarding query results whose top-j covers at least one of the subtopics well. Our second formulation, coined DELTA, derives per-rank gain values from the observed differences in per-subtopic gain values as

$$g(j) = \max \left(0, \max_{1 \leq i \leq |q|} (g(i,j) - g(i,j-1)) \right) . \tag{3.10}$$

Note that the outer maximum function in Equation 3.10 is to guarantee that $g(j) \geq 0$. For a query result to obtain a high per-rank gain value under this formulation, its result document at rank j must be closely related to a subtopic that has not yet been covered.

Position bias. As a final step, we describe how the per-rank gain values $g(j)$ defined above can be aggregated into a single measure reflecting the quality of a top-k result. By definition, as described above, in our approach the influence of documents at lower ranks is diminishing. Thus, as a first formulation that we explore, we simply sum up the per-rank gain values observed at ranks up to k as

$$\sum_{1 \leq j \leq k} g(j) . \tag{3.11}$$

In the following, this formulation is referred to as NB for no bias. We also consider a second alternative formulation, coined RB, that borrows the position-bias model from Rank-Biased

		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
$\frac{KLD(\Theta_{q_i} \parallel \Theta_{R_k})}{KLD(\Theta_{q_i} \parallel \Theta_{\mathcal{D}})}$	Subtopic 1	.49	.29	.29	.27	.28	.29	.28	.27	.28	.29
	Subtopic 2	.41	.42	.50	.54	.60	.65	.67	.70	.75	.79
	Subtopic 3	.79	.51	.53	.54	.25	.26	.23	.23	.23	.23
	Subtopic 4	.72	.67	.77	.80	.86	.92	.93	.94	.99	1.00
$g(i,k)$	ABS	.60	.71	.71	.73	.75	.74	.77	.77	.77	.77
	DELTA	.60	.28	.00	.02	.29	.00	.03	.00	.00	.00

Table 3.2: Example based on a result ranking from TREC WebTrack 2012

Precision (Moffat and Zobel, 2008) and aggregates per-rank gain values as

$$(1-\theta) \cdot \sum_{1 \leq j \leq k} g(j) \cdot \theta^{j-1} \,. \tag{3.12}$$

The parameter θ (set as $\theta = 0.8$ in our experiments) models the user's persistence in sifting through the query result, or put differently, at each rank the user decides to stop inspecting query results with probability $(1-\theta)$.

Example. To ease the understanding of our methods, we use a concrete example. We pick up a result ranking for query with 4 subtopics from TREC WebTrack 2012. We thus obtain four language models Θ_{q_i}, each estimated from the corresponding relevant documents. When evaluating the obtained query result at depth 10, at each position we compute the divergence $\frac{KLD(\Theta_{q_i} \parallel \Theta_{R_k})}{KLD(\Theta_{q_i} \parallel \Theta_{\mathcal{D}})}$. For instance, when computing $\frac{KLD(\Theta_{q_1} \parallel \Theta_{R_3})}{KLD(\Theta_{q_1} \parallel \Theta_{\mathcal{D}})}$, we use documents d_1, d_2, d_3 to estimate a language model and compare it with Θ_{q_1}. Based on the divergence, we further compute the aforementioned two types of gain. We first convert the divergence according to Equation 3.8. For ABS, we get the maximum value among the 4 subtopics, and for DELTA, we pick up the maximum of the delta value as $g(i,k)$ at each position k. The results are summarized in Table 3.2.

3.6.2 Result

In this section, we design experiments to investigate the reliability of established measures, and to examine the proposed measures under three different aspects: (i) the robustness when only few judgments available, (ii) how well they can reuse relevance judgments to evaluate systems on a previously unseen document collection, and (iii) their correlation with existing cascade measures.

Cascade Measures. As cascade measures that we compare against, we consider α-nDCG, ERR-IA, and NRBP, which are introduced in Section 2.2.4. Combining the different design choices regarding per-rank gain and position bias, we obtain four novel measures as follows.

- AbsNb. Combination of Abs gain from Equation. 3.9, and no position bias, coined as Nb, as in Equation. 3.11;
- AbsRbp. Combination of Abs gain from Equation. 3.9, and Rb ranking bias as in Equation. 3.12;
- DeltaNb. Combination of Delta gain from Equation. 3.10, and Nb bias as in Equation. 3.11;
- DeltaRbp. Combination of Delta gain from Equation. 3.10, and Rb bias as in Equation. 3.12.

Rank Correlation. We use Kendall's τ as introduced in Section 2.2.4. Note that, compared with the task in Section 3.5, we deal with a much harder task due to the consideration of diversify and novelty which makes the problem more complicated, and to the settings where only very few judgments are available. Voorhees suggested 0.9 as a threshold to consider two rankings as equivalent, whereas a correlation below 0.8 reflects a significant difference (Voorhees, 2001). In this section, however, we choose 0.8 as a threshold given the difficulty of the task itself.

Aspects that we examine in our experiments are: (i) *Robustness* in the presence of incomplete judgments by removing most of relevance judgments and comparing against the system rankings determined by the established cascade measures on complete judgments; (ii) *Reusability* of relevance judgments by evaluating measures in terms of their ability to rank systems operating on CwB based on relevance judgments collected on the disjoint document collection CwC, which is introduced in Section 3.3; (iii) *Correlation* with the established cascade measures in terms of Kendall's τ between the obtained system rankings to see how closely our measures approximate those.

Robustness over Sparse Judgments

Firstly, we analytically investigate the effectiveness of different cascade measures when evaluating the raw list and the condensed list as in (Sakai et al., 2012) of search results. Beyond that, we evaluate the proposed measures under the same set of judgments and make comparisons. In particular, we inspect the correlation between system rankings determined by different measures on incomplete judgments and the ones determined by established cascade measures over complete judgments. To do so, we employ similar procedure employed in (Buckley and Voorhees, 2004) and (Bompada et al., 2007), constructing incomplete judgments in an analytical way. We denote the full relevance judgments documents set as qrel. Given a query, we randomly shuffle the relevant documents in qrel, and pick up first $max(1, \lceil p\%|qrel|\rceil)$ relevant documents from qrel to build the Θ_q. We test two random sampling settings. The first one is based on queries, that is, for each query we require at least one relevant document to construct our measures. Another is based on subtopics, i.e., at least one relevant document is required for each subtopic. The difference between these two sampling strategies is that, many subtopics only contain one relevant document in the complete qrel, therefore it is very probable there existing subtopics with no relevant document if under the query-based sampling, especially with small p. These relevant judgments are all that is required by our proposed measures. To compare with results based on established measures over condensed lists, we further sample p% non-relevant documents to construct an incomplete judgments including p% judgments. To remove the randomness from this sampling procedure, we report the average results from 30 repetitions. Though this stratified random sampling is analytical, it can cover different situations through dozens of samplings.

Kendall's τ correlations relative to the ones under complete judgments with ERR-IA, α-nDCG and NRBP are summarized in Figures 3.10, and 3.11 and 3.12 respectively, when using query-based sampling. Whereas Figures 3.13, and 3.14 and 3.15 show the results when using subtopic-based sampling. The two dashed curves in these figures represent the system rankings determined by established measures, namely, ERR-IA, α-nDCG and NRBP, when measuring on raw lists and condensed lists respectively. Based on pilot experiments, we only display results for AbsRb and DeltaRb in this experiment, denoted as two solid curves, since the other two behave similarly. The x-axis indicates the sample percentage p%, and the y-axis is the Kendall's τ correlation. Note that, akin to the discussions in Section 3.5.2, one can not guarantee that more judgments can always lead to a higher correlation when very few judgments are available, as indicated on WT2012 in Figure 3.10. This is because when few judgments are available the reliability of the measure computation highly depends on the particular subset of documents being judged.

Under query-based sampling, it can be seen that these established measures require more than 40%-50% judgments to achieve 0.8 Kendall's τ correlation; whereas under subtopic-based sampling, 30%-40% judgments are required. Sakai et al. demonstrated that, the condensed-list methods can address the incomplete judgments issues in leave-one-out experiment (Sakai et al., 2012). However, it is clear from Figure 3.10, and 3.11 and 3.12 and from Figure 3.13, and 3.14 and 3.15, with sparse judgments, namely when less than 30% judgments available, the correlation for condensed lists can be very low, e.g., smaller than 0.4 with less than 1% judgments available. This is not surprising in the sense that the sparse judgments make the computation of the established measures highly depend on the very few documents being labeled. Put differently, an unjudged document directly corresponds to a missing component in the formula of these measures. Meanwhile, the established measures behave much more smoothly, given the fact that the document contents, instead of the judgments are directly consumed. In other words, even with a single judged relevant documents, one can still estimate a reasonable language model as Θ_q out of it, given that documents that are relevant to the same query tend to be similar in the content. As a concrete number, the correlation numbers for the established measures vary a lot among different years. Observe that DeltaRb still obtains a Kendall's τ correlation above 0.8 for the year 2011 with as little as 15% of relevance judgments. Likewise, for the year 2012 we observe Kendall's τ correlation above 0.8 with as

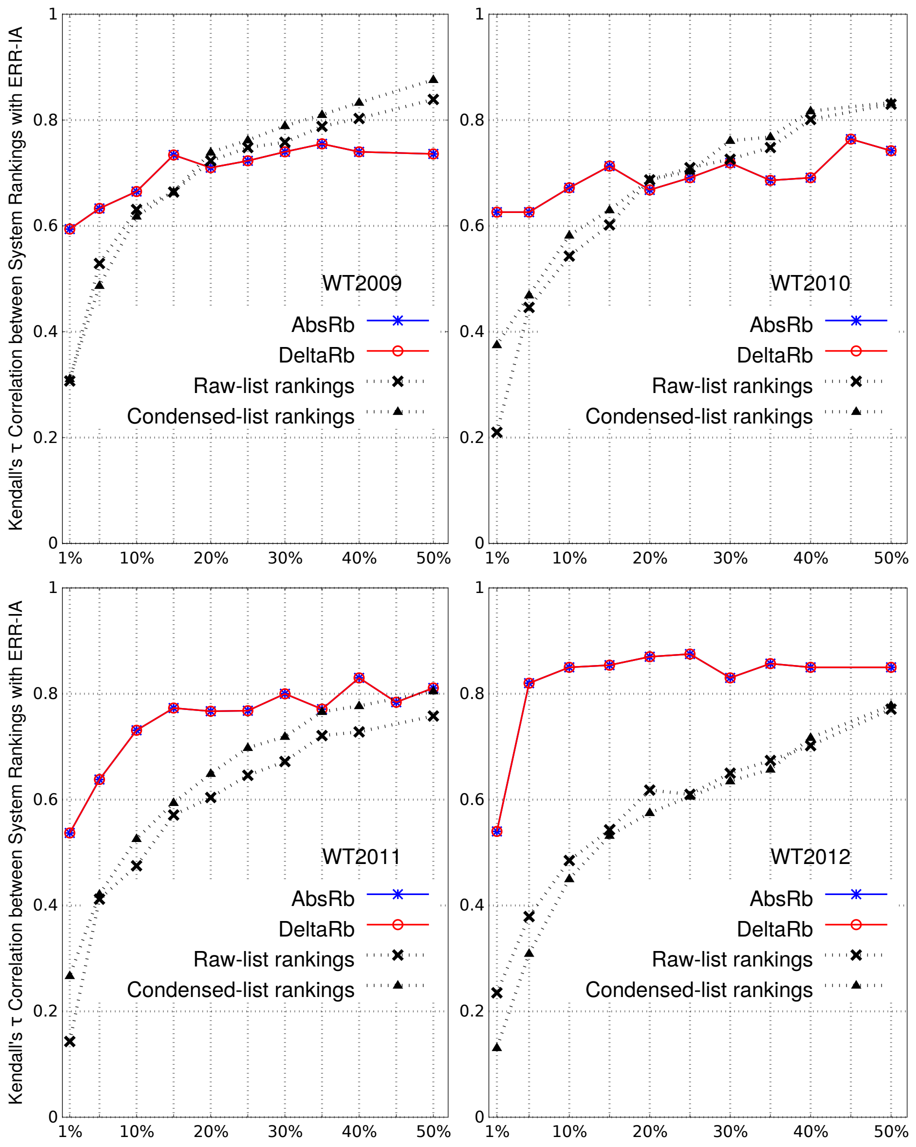


Figure 3.10: Effectiveness of cascade measures over the raw list and the condensed list when examining with query-based sampling relative to **ERR-IA**. TREC Web Track 2009–2012 are used. In each figure, the x-axis indicates the sampling percentage $p\%$ and the y-axis indicates the Kendall's τ correlation.

little as 5% relevance judgments. Meanwhile, on 2009 and 2010, the proposed measures fail to reach beyond 0.8 Kendall's τ, but achieve significantly higher correlation than established measures when less than 15% judgments are available. Note that, different from the established measures, both DeltaRb and AbsRb behave rather robust when different amount of judgments are available, and the correlation values do not increase monotonically. This is due to the fact that more judgments can only adjust Θ_q by including more observations of the distribution, which is fundamentally different from the way when computing established measures by taking individual relevant judgments into computation. Finally, we argue that there is no significant difference

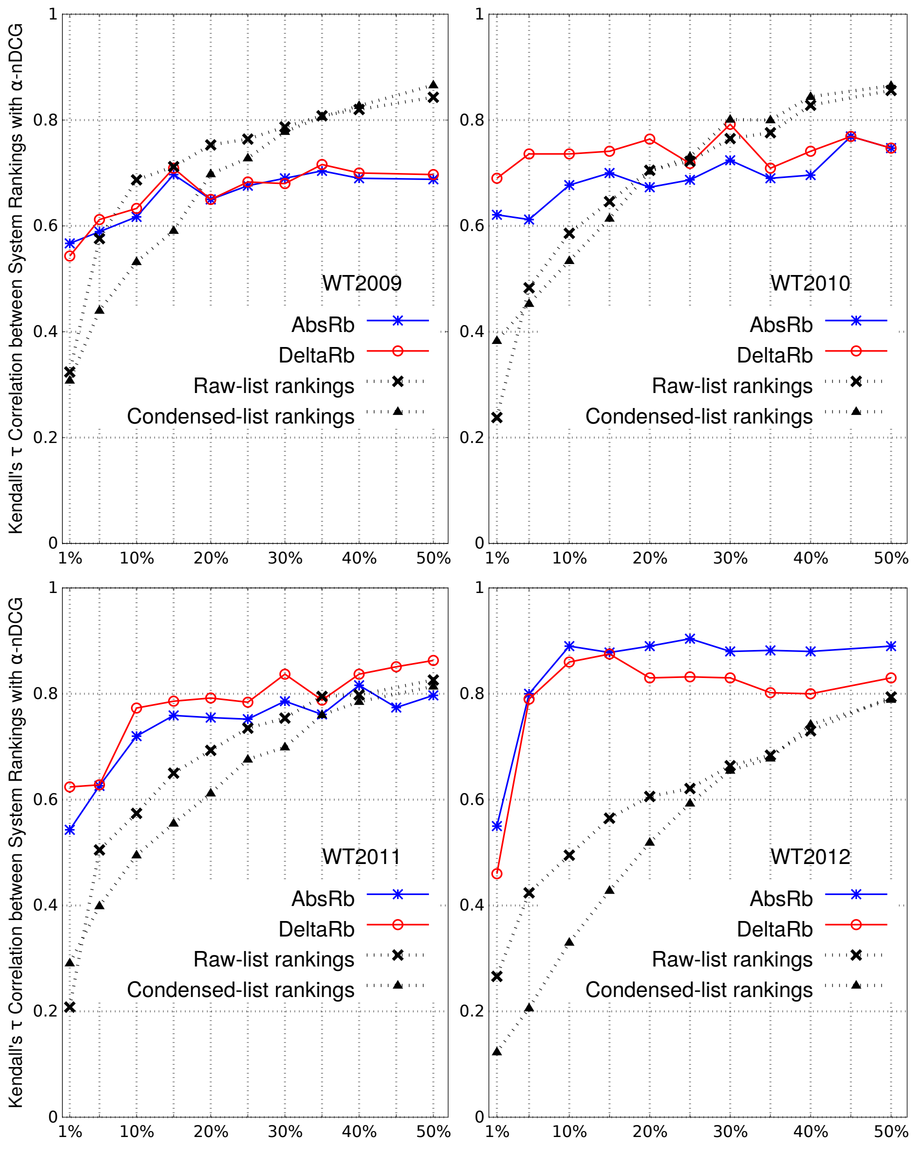


Figure 3.11: Effectiveness of cascade measures over the raw list and the condensed list when examining with query-based sampling relative to α-**nDCG**. TREC Web Track 2009–2012 are used. In each figure, the x-axis indicates the sampling percentage p% and the y-axis indicates the Kendall's τ correlation.

between query-based and subtopic-based sampling according to the figures.

Reusability on Disjoint Document Collection

As a second aspect, we further examine whether our measures are able to reuse relevance judgments collected on one document collection to evaluate systems on another (disjoint) document collection. Note that this setting is different from when $p = 0\%$, which corresponds to having no relevance judgments available at all and is beyond hope for any measure. Instead, we estimate

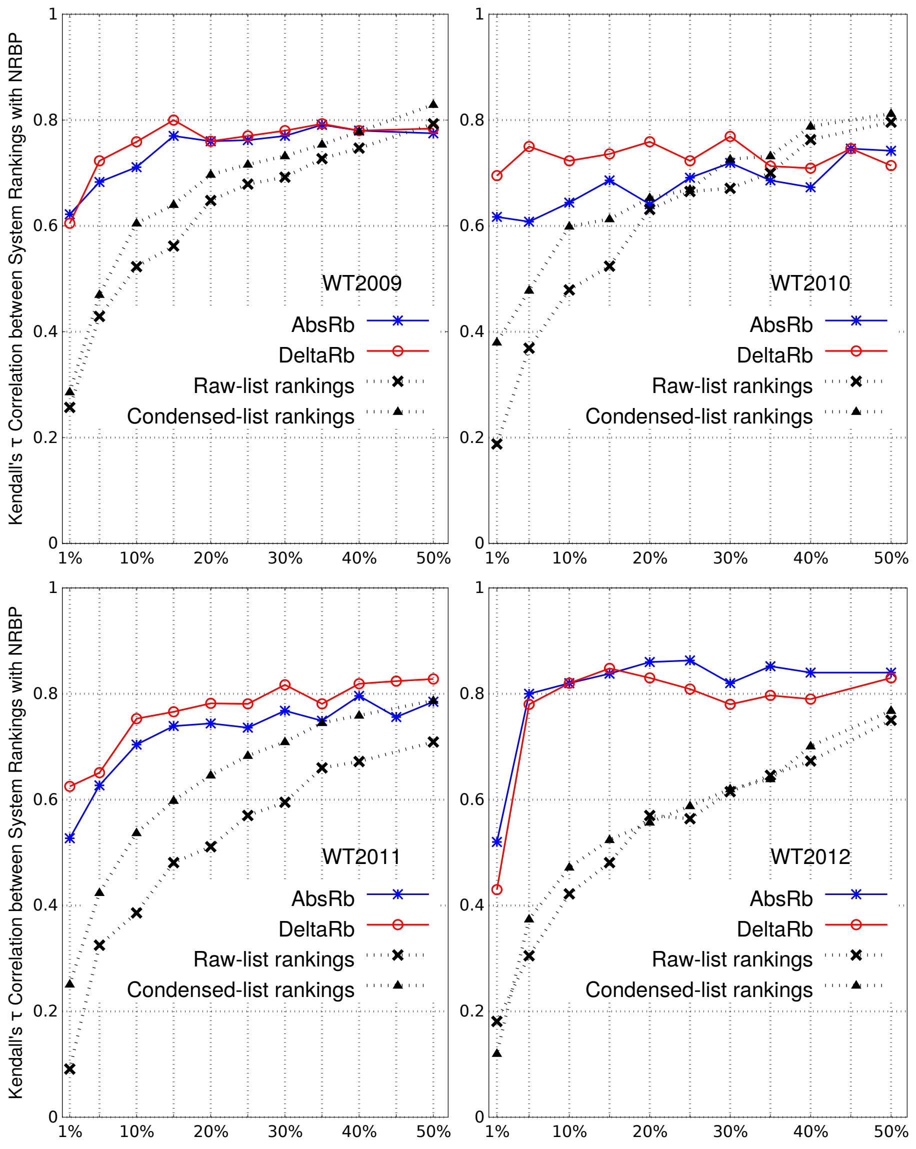


Figure 3.12: Effectiveness of cascade measures over the raw list and the condensed list when examining with query-based sampling relative to **NRBP**. TREC Web Track 2009–2012 are used. In each figure, the x-axis indicates the sampling percentage $p\%$ and the y-axis indicates the Kendall's τ correlation.

subtopic language models based on documents from CWC. Our objective is then to approximate the system ranking determined by ERR-IA and α-nDCG on CWB, which by construction is disjoint from CWC (we recall that CWC is the set of documents that appear in CWA but not in CWB). In this context, it is worth mentioning that CWB, despite of its smaller size, comes with 1.5× more relevance judgments than CWC, which is due to the facts that a lot of systems opt for CWB to work on. Table 3.3 reports the obtained Kendall's τ correlations. It can be seen that DELTARB performed better among the proposed measures. Though only in 2011 over 0.8 correlation can be achieved, on other years the correlation is beyond 0.5, and in 2009-10, the correlation is around

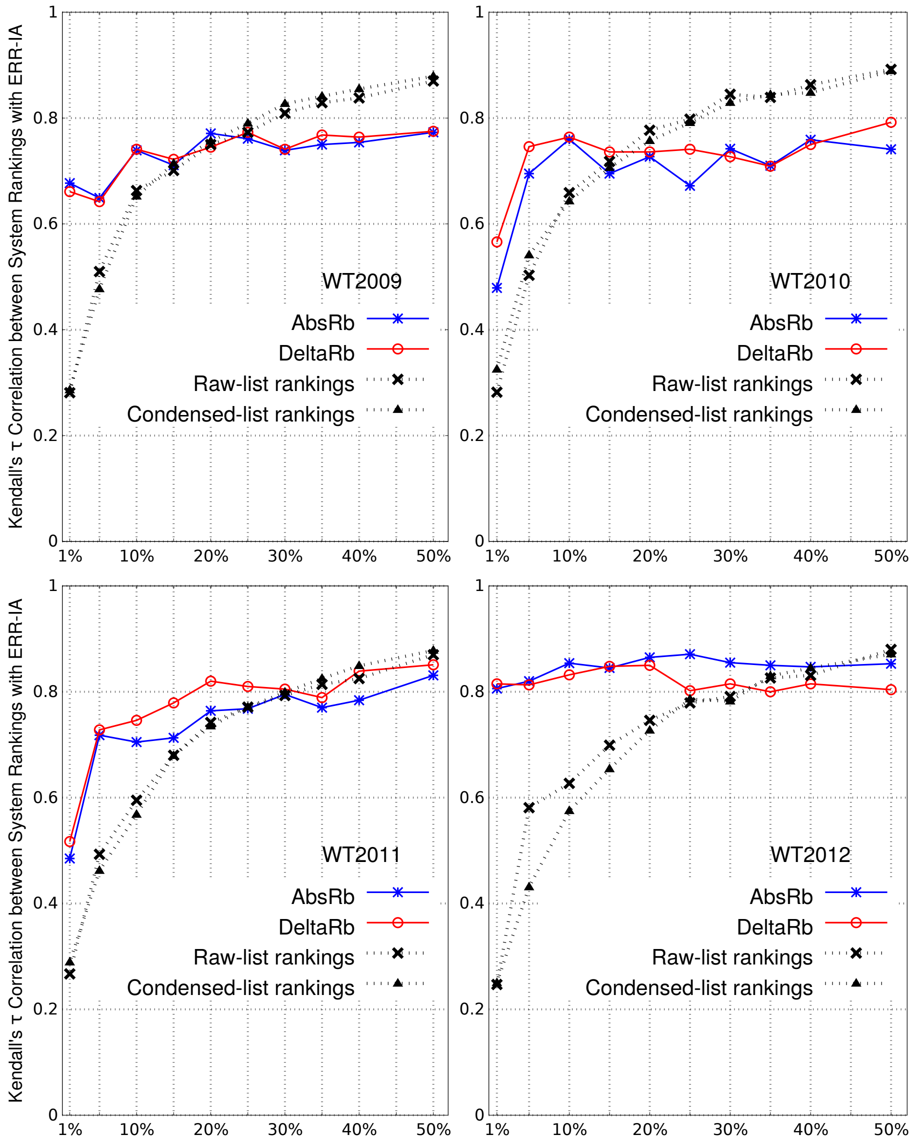


Figure 3.13: Effectiveness of cascade measures over the raw list and the condensed list when examining with subtopic-based sampling relative to **ERR-IA**. TREC Web Track 2009–2012 are used. In each figure, the x-axis indicates the sampling percentage $p\%$ and the y-axis indicates the Kendall's τ correlation.

0.75. Note that established cascade measures, in contrast, can not be employed in this setting with a complete mismatch between relevance judgments and result documents. This actually highlights the advantages of the proposed measures in fully utilizing judged documents that do not appear in the evaluated ranking of documents.

In Table 3.3, for the year 2012 we observe a relatively low value. Digging deeper we want to investigate the question to what extent reusability depends on the document collection on which relevance judgments were collected. Therefore, for the year 2012, we further employ all our document collections CwA, CwB, and CwC as a source of relevance judgments and study

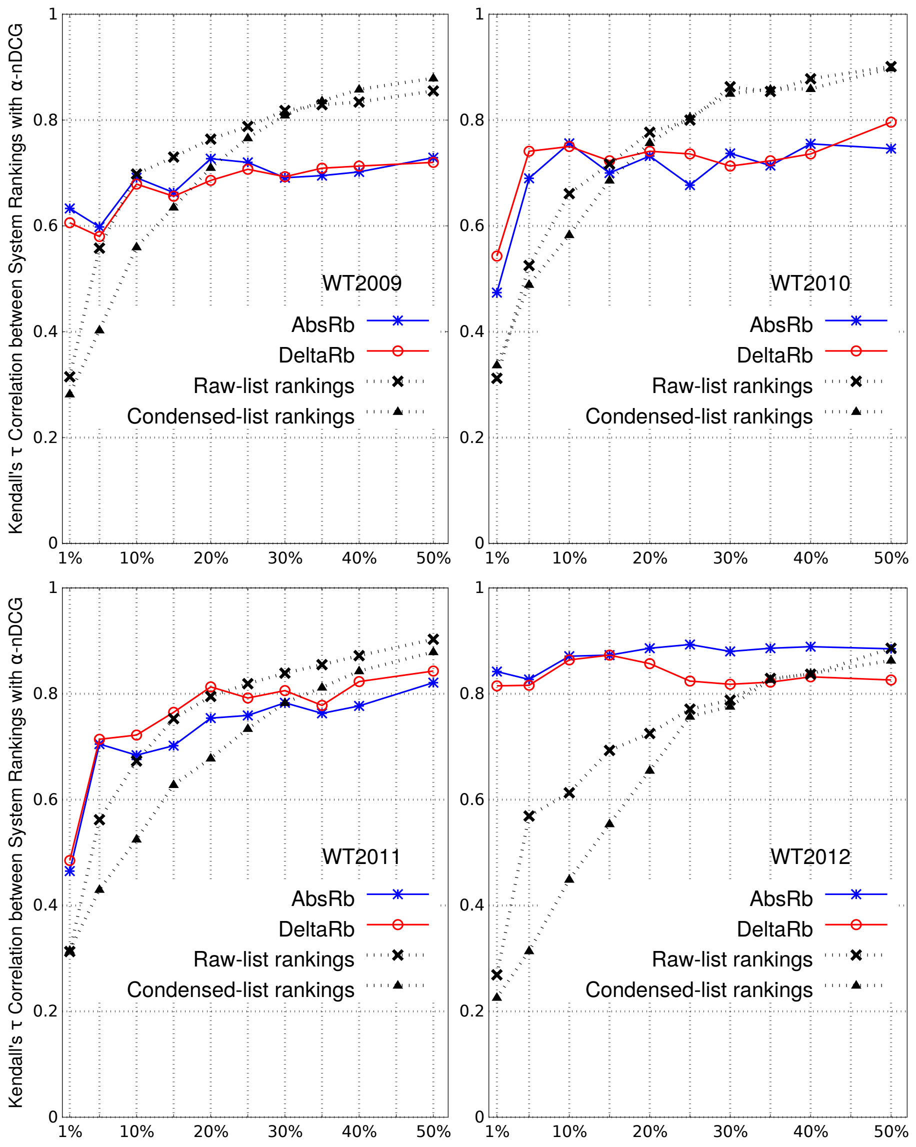


Figure 3.14: Effectiveness of cascade measures over the raw list and the condensed list when examining with subtopic-based sampling relative to α-**nDCG**. TREC Web Track 2009–2012 are used. In each figure, the x-axis indicates the sampling percentage $p\%$ and the y-axis indicates the Kendall's τ correlation.

correlation with α-nDCG, ERR-IA, and NRBP on all these three document collections. Recall that CwC is disjoint from CwB, while both CwB and CwC are subsets of CwA. Table 3.4 shows Kendall's τ correlations for all combinations of document collections. From the table we can see that the choice of document collection on which relevance judgments are collected can have a significant impact. Thus, Kendall's τ correlations are generally higher for relevance judgments collected on CwA and CwB than on CwC. This is not completely surprising, given that many participants of TREC Web Track 2009-2012 initially focused on CwB, and CwC was constructed artificially as mentioned. What is promising is that using relevance judgments from CwB to

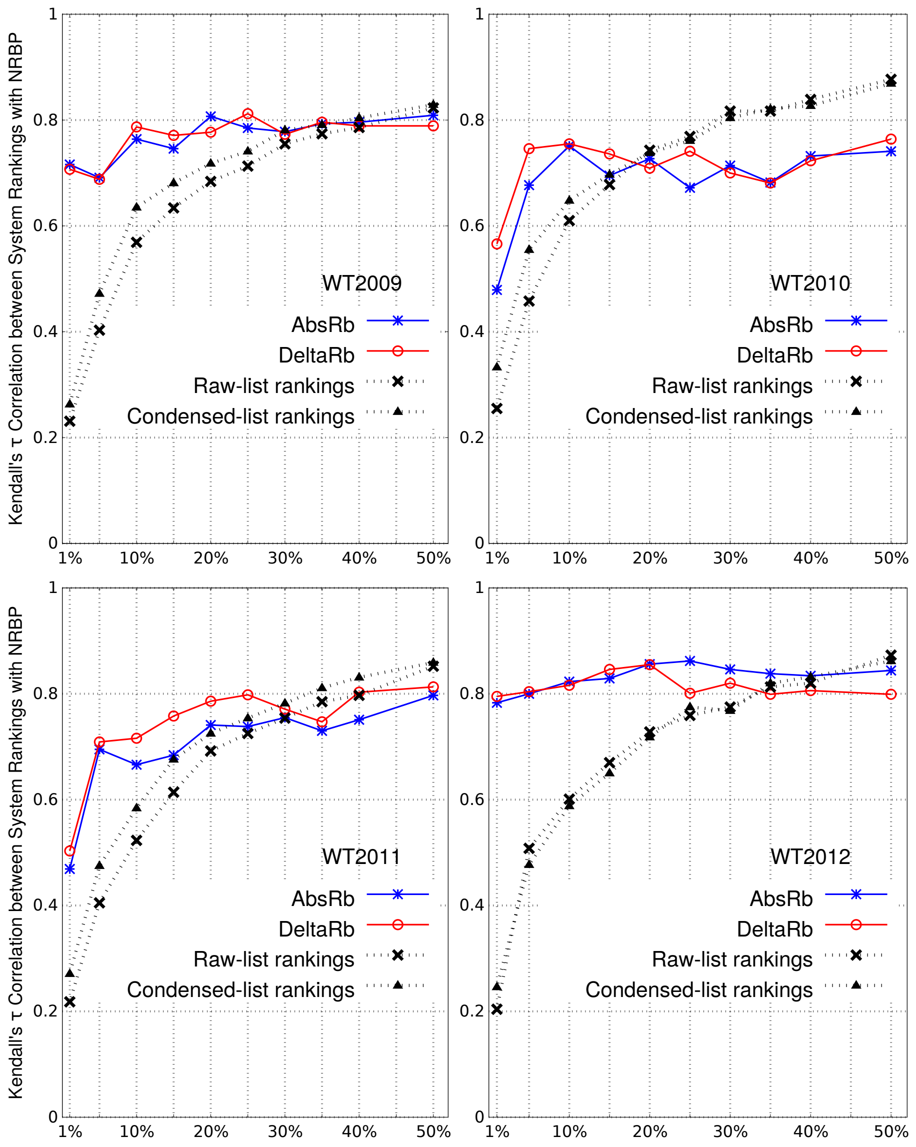


Figure 3.15: Effectiveness of cascade measures over the raw list and the condensed list when examining with subtopic-based sampling relative to **NRBP**. TREC Web Track 2009–2012 are used. In each figure, the x-axis indicates the sampling percentage $p\%$ and the y-axis indicates the Kendall's τ correlation.

evaluate systems on the much larger CWA works fine when using our measures, as can be seen from the fact that observed values of Kendall's τ decrease only slightly if at all. It is also worth mentioning that we performed analogous experiments for the years 2009–2011 and with similar observations which we hence omit here.

		AbsNb	AbsRb	DeltaNb	DeltaRb
	α-nDCG	.72	.70	.73	.69
2009	ERR-IA	.78	.78	.76	.76
	NRBP	.74	.74	.70	.74
	α-nDCG	.72	.66	.74	.76
2010	ERR-IA	.70	.66	.72	.75
	NRBP	.68	.65	.71	.73
	α-nDCG	.71	.76	.79	**.81**
2011	ERR-IA	.67	.75	.73	**.81**
	NRBP	.64	.74	.69	.79
	α-nDCG	.23	.40	.31	.51
2012	ERR-IA	.26	.44	.31	.54
	NRBP	.26	.45	.31	.54

Table 3.3: Reusability of our measures. Relevance judgments collected on CwC are used to evaluate systems on the disjoint document collection CwB. Kendall's τ above 0.8 shown in bold.

Correlation

Finally, we examine the correlation between the proposed measures and the established cascade measures. Though the proposed measures aim at addressing the cases when only very sparse judgments are available, one may desire to know the relationship between the proposed measures and the established ones. To this end, we compute Kendall's τ between the system rankings determined by our measures and the ones determined by the established cascade measures.

For comparison, Table 3.5 lists pairwise correlations between α-nDCG, ERR-IA, and NRBP in terms of their average Kendall's τ on TREC Web Track 2009–2012. It is apparent from the table that the established cascade measures are highly correlated.

Table 3.6 reports Kendall's τ between our four measures AbsNb, AbsRbp, DeltaNb and DeltaRbp, and the cascade measures α-nDCG, ERR-IA, and NRBP. For a different perspective, Figure 3.16 plots system ranks assigned by our four methods against those assigned by ERR-IA on TREC Web Track 2009–2012. For a measure having perfect correlation with ERR-IA points in this plot would lie on the main diagonal $y = x$ line. We show plots against ERR-IA here. From Table 3.6, we can see that the correlation between our measures and α-nDCG, ERR-IA, and NRBP varies across different query sets. The correlation is lowest for queries from the year 2010 and highest for queries from the year 2012. Comparing different methods, we observe a positive

		CwA			CwB			CwC		
		CwA	CwB	CwC	CwA	CwB	CwC	CwA	CwB	CwC
$\alpha-$nDCG	AbsNb	**.87**	**.89**	.71	.79	**.90**	.73	.30	.23	.71
	AbsRb	**.89**	**.89**	.71	**.84**	**.89**	.73	.43	.40	.71
	DeltaNb	.71	.79	.67	.61	**.81**	.67	.23	.31	.69
	DeltaRb	**.82**	**.88**	.69	.79	**.88**	.67	.41	.51	.70
ERR-IA	AbsNb	**.85**	**.87**	.65	**.81**	**.88**	.69	.29	.26	.67
	AbsRb	**.86**	**.88**	.69	**.85**	**.89**	.71	.43	.44	.68
	DeltaNb	.69	.75	.63	.60	.78	.60	.20	.31	.65
	DeltaRb	**.80**	**.86**	.65	**.80**	**.87**	.64	.41	.54	.66
NRBP	AbsNb	**.84**	**.83**	.60	**.82**	**.83**	.64	.26	.26	.63
	AbsRb	**.85**	**.84**	.64	**.86**	**.85**	.66	.40	.45	.63
	DeltaNb	.68	.70	.59	.61	.72	.55	.19	.31	.60
	DeltaRb	**.80**	**.82**	.60	**.82**	**.84**	.59	.39	.54	.61

Table 3.4: Impact of document collection used for collecting relevance judgments. The first row indicates the document collection on which relevance judgments were collected; the second row indicates the document collection on which query results were determined. Kendall's τ correlations above 0.8 are shown in bold.

effect of the position bias with AbsRb and DeltaRb consistently showing higher correlation than their non-biased counterparts. While our measures do not consistently achieve a Kendall's τ correlation above 0.8, we argue that the proposed measures are still useful, since they can better deal with incomplete judgments and more effectively reuse relevance judgments, as we discussed. From Figure 3.16, it can be seen that all points distribute along the $y = x$ line, indicating that the system rankings determined by the proposed measures are close to the ones from ERR-IA. Moreover, in all years, it is clear that the points locate in the upper right corner, corresponding to results from top-ranked systems, distribute more on the $y = x$. This demonstrates that the proposed measures agree better with ERR-IA for the top-ranking systems, but disagree more for the systems that ranked in the middle.

3.7 Conclusion

Low-cost evaluation has been an active area of research within information retrieval for the past decades.

In this chapter, we first investigated the agreement to the *cluster hypothesis* when employing different document representations, and demonstrate that the traditional **bag-of-words sparse**

	α-nDCG	ERR-IA	NRBP
α-nDCG			
ERR-IA	.93		
NRBP	.88	.87	

Table 3.5: Average Kendall's τ between α-nDCG, ERR-IA, NRBP on TREC Web Track 2009–2012.

		AbsNb	AbsRb	DeltaNb	DeltaRb
2009	α-nDCG	.78	.70	.78	.73
	ERR-IA	**.81**	.75	.79	.78
	NRBP	**.80**	.79	.73	.78
2010	α-nDCG	.70	.73	**.81**	.76
	ERR-IA	.68	.73	.77	.76
	NRBP	.65	.71	.75	.72
2011	α-nDCG	.74	**.81**	.76	**.85**
	ERR-IA	.74	**.81**	.75	**.85**
	NRBP	.71	.78	.70	**.81**
2012	α-nDCG	**.87**	**.89**	.71	**.82**
	ERR-IA	**.85**	**.86**	.69	**.80**
	NRBP	**.84**	**.85**	.68	**.80**

Table 3.6: Correlation with established cascade measures. Kendall's τ correlations above 0.8 shown in bold.

vector with tf-idf weighting performs good on different benchmarks relative to different more advanced representations.

We also investigated how different strategies for selective labeling and mitigating incomplete labels interact, and demonstrated that label prediction is a robust and viable strategy to mitigate incomplete labels, as long as at least 20% of documents have been labeled as training data. Moreover, with label prediction in mind, we proposed a novel strategy MaxRep for selective labeling. In contrast to existing strategies, it considers both ranking information and document contents and seeks to select a representative subset of documents to label. Our experiments confirmed that MaxRep is beneficial and outperforms other strategies when label prediction is used.

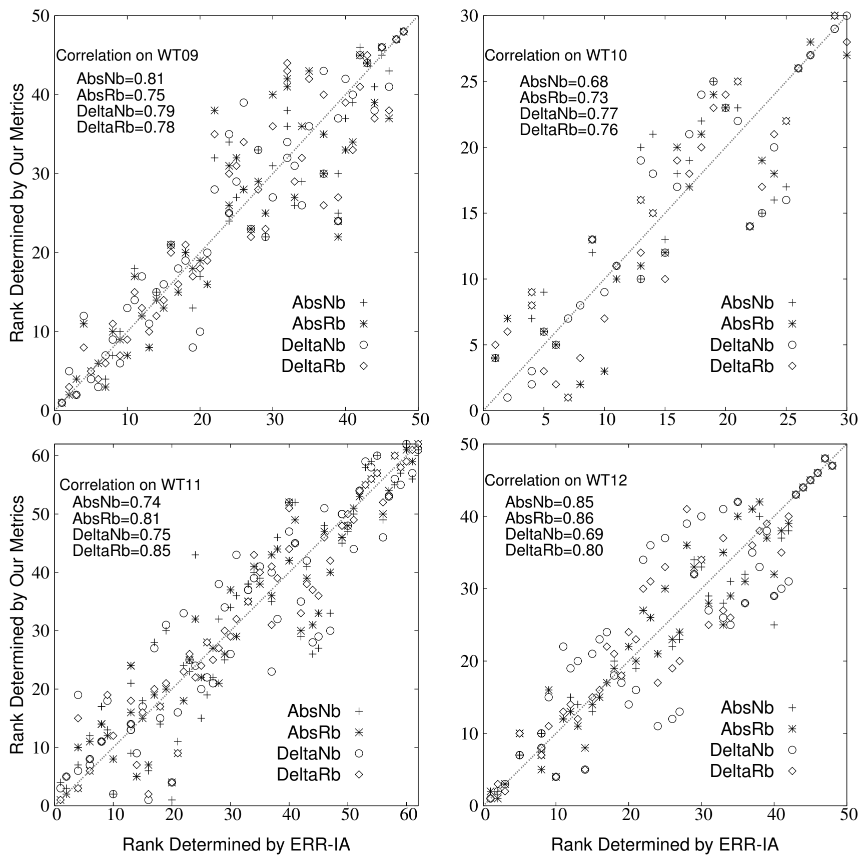


Figure 3.16: System rank assigned by ERR-IA vs. system rank assigned by our measures on TREC Web Track 2009–2012.

Finally, we investigated the performance of the established cascade measures α-nDCG, ERR-IA, and NRBP under incomplete judgments. We find that their ability to rank systems reliably deteriorates quickly as we remove more and more relevance judgments. To mitigate this, we propose novel cascade measures that are based on the Kullback-Leibler divergence between language models estimated for subtopics and returned query results. Our experiments showed that our novel measures correlate sufficiently with the established ones and –more importantly– are robust when faced with incomplete judgments. Even on as little as 15% of relevance judgments, our novel cascade measures still get close to the established ones on complete judgments. Another benefit of our novel measures is that they can reliably reuse judged documents that are not included in the evaluated ranking, utilizing the expensive manual judgments more effectively. This property particularly make the reusability of judgments across document collections possible, as demonstrated in our experiments, whereas established cascade measures fail to work in this case.

4 Low-cost Evaluation with Preference Judgments

4.1 Introduction

Preference judgments have been demonstrated as a better alternative to the widely used graded judgments. Compared with graded judgments, preference judgments lead to better inter-assessors agreement, less time consumption per judgment (Carterette et al., 2008) and better judgment quality in terms of agreement to user clicks (Kazai et al., 2013). As pointed out in (Radinsky and Ailon, 2011), these advantages come from the pairwise nature of preference judgments, namely, the documents in the pair can mutually act as a “context”, providing a reference for the judges. Especially, Kazai et al. demonstrated that preference judgments collected using crowdsourcing can be inexpensive yet high-quality. In their experiments preference judgments yielded good quality, getting close to the ones obtained from trained judges in terms of user satisfaction. As mentioned in Section 2.2.2, crowdsourcing makes the judgments scalable and cheaper, compared with collecting judgments from trained assessors.

Unfortunately, preference judgments are very expensive in terms of the number of judgments. And this is still true when using crowdsourcing. To judge the relevance of N_d documents, $\mathcal{O}(N_d^2)$ preference judgments are needed, since pairs of documents have to be considered, whereas $\mathcal{O}(N_d)$ graded judgments suffice. This means that collecting preference judgments for N_d documents requires $(N_d^2 - N_d)/2$ judgments as in (Kazai et al., 2013) and (Radinsky and Ailon, 2011). Luckily, it has been shown that preference judgments are transitive in (Carterette et al., 2008) and (Rorvig, 1990) when collected from trained judges, which can be exploited to reduce their required number to $\mathcal{O}(N_d \log N_d)$. Beyond that, previous works have considered different variants of preference judgments. When judges are asked to state strict preferences for two documents d_1 and d_2, as done in (Carterette et al., 2008), (Radinsky and Ailon, 2011) and (Rorvig, 1990), they can only indicate whether d_1 is preferred over d_2 ($d_1 \succ d_2$) or vice versa ($d_1 \prec d_2$). When asking for weak preferences, additional options are provided, allowing judges to state that the two documents are tied ($d_1 \sim d_2$) as in (Kazai et al., 2013), (Song et al., 2011) and (Zhu and Carterette, 2010);

or two documents are either equally relevant or equally non-relevant (Bashir et al., 2013). We argue that allowing for ties is natural when judging search relevance, since it is unlikely that each of the possibly hundreds of returned documents has its own degree of relevance. Assuming transitivity, we first demonstrate that the number of judgments can be reduced dramatically. Moreover, we try to answer "whether one can assume transitivity when collecting judgments from crowdsourcing?"

Another difference between graded judgments and preference judgments, as reported in (Carterette et al., 2008), is that preference judgments tend to be less time consuming. In their experiments, trained judges took 40% less time to make individual preference judgments than to make individual graded judgments. We investigate whether this observation also holds when judgments are collected using crowdsourcing. If so, there is an opportunity to reduce cost by paying less for preference judgments.

Thus we firstly answer the following questions. Thereafter exploring the usage of ties to reduce the number of judgments. Finally, we combine the empirical results with the ties and propose a low-cost preference judgment method.

RQ1: *Whether ties can be used to reduce the number of judgments?*

RQ2: *Can weak/strict preference judgments collected using crowdsourcing replace judgments by trained judges in* TREC*?*

RQ3: *Do weak/strict preference judgments exhibit transitivity when collected using crowdsourcing?*

RQ4: *How do weak/strict preference judgments compare against graded judgments in terms of time consumption?*

For **RQ1**, we investigate through analysis and empirical studies to demonstrate the potential of ties in reducing the number of judgments when transitivity is strictly observed, as in (Carterette et al., 2008), (Niu et al., 2012) and (Song et al., 2011). Different from existing works, we focus on the relationship between the introduction of ties and the number of judgments. For **RQ2**, Kazai et al. demonstrated that the weak preference judgments collected from crowdsourcing already achieve judgment quality close to the one from the graded judgments collected from trained judges. The click data is employed to quantify judgments in (Kazai et al., 2013). In this work, we try to re-examine the conclusion by directly comparing the collected judgments with the ones from TREC as in (Alonso and Mizzaro, 2009). Beyond that, we examine both strict and weak judgments in terms of their quality, investigating whether the two kinds of preference judgments are the same in this regard. For **RQ3**, firstly, whether transitivity still holds when preference judgments are collected using crowdsourcing is an open question as mentioned in (Bashir et al., 2013). In the aforementioned studies (Carterette et al., 2008) and (Rorvig, 1990), trained judges stated their relative preference for all pairs of documents returned for a specific query. As a consequence, when considering a triple of documents, the same judge states relative preferences for all pairs of documents therein, making transitivity more a matter of judges' self-consistency. When using crowdsourcing, in contrast, it is very unlikely that the same judge states relative preferences for all pairs of documents from a triple, given that workers typically only contribute a small fraction of work. Transitivity, if it exists, can thus only be a result of agreement among different judges. We examine whether transitivity holds when preference judgments are collected using crowdsourcing, when considering preference judgments aggregated from the stated preferences of multiple different judges. In addition, though transitivity is examined among strict preferences in (Carterette et al., 2008), the transitivity among weak preference judgments have never been tested. For **RQ4**, as mentioned, Carterette et al. demonstrated that trained judges tended to consume less time when making preference judgments. We investigate whether this observation remains true when using crowdsourcing.

To answer **RQ1**, we reexamine the number of preference judgments on N_d documents with established *Quick-Sort-Judge* mechanism from (Song et al., 2011). Moreover, we empirically investigate the number of judgments when simulating the ground-truth ranking from TREC Web Track 2011-2014. We argue that the tie is a compromise between the number of judgments and the judgment granularity.

We demonstrate that, with transitivity, ties actually cluster documents into tie partitions, and reduce the ranking of documents to the ranking of tie partitions and we demonstrate that the average number of judgments is reduced to $\mathcal{O}(N_t \log N_d)$, where N_t is the number of tie partitions. Beyond that, we also demonstrate that, it is even possible to further reduce this number to $\mathcal{O}(N_t \log N_t + N_d)$, by tuning the judgment mechanism to leverage the "cluster effects" from ties.

To answer **RQ2–RQ4**, we conduct an empirical study on CrowdFlower[1]. Using topics and pooled documents from the TREC Web Track,[2] we collect graded judgments, weak preference judgments, and strict preference judgments. We assess the inter-judge agreement for the different kinds of judgments and also examine to what extent they can replace judgments by trained judges from TREC. Akin to (Carterette et al., 2008), we examine transitivity by considering triples of documents. To analyze the time consumption for different kinds of judgments, our user interface is carefully instrumented to record the time that it takes judges to read documents and to make their judgments.

We see that preference judgments collected using crowdsourcing tend to show better agreement with TREC judges. Moreover, the agreement between strict preference judgments from crowdsourcing and judgments from TREC already match the agreement among trained judges reported

[1]http://www.crowdflower.com/
[2]http://trec.nist.gov/data/webmain.html

from literature (Carterette et al., 2008) and (Kazai et al., 2013). In addition, from our empirical study using crowdsourcing, we observe that transitivity holds over 90% for strict preference judgments collected using crowdsourcing; for weak preference judgments it only holds for about 75% of triples. Finally, we find that judges spend more time when asked for preference judgments than graded judgments in terms of total time consumption. Though time on making a single judgment is found to be lower for strict preference judgments.

4.2 Related Work

Preference judgment is better. In early work, Rorvig proposed that the preference judgments could be used instead of the scaling-based judgments due to the applicability of simple scalability on documents. Since then, the advantages of preference judgments over graded judgments have been empirically tested and confirmed in (Carterette et al., 2008), (Radinsky and Ailon, 2011) and (Song et al., 2011). Preference judgments have been demonstrated as a better alternative to graded judgments, since there is no need to define graded levels (Carterette et al., 2008), their higher inter-assessor agreement and better quality according to (Kazai et al., 2013) and (Radinsky and Ailon, 2011).

Weak preferences versus strict preferences. The choice between two kinds of preferences varied a lot among different works, even though some of them share similar motivations or research methodologies. Carterette et al., Radinsky and Ailon, as well as Rorvig employed strict preferences in their empirical studies for preference judgments. In the meantime, Kazai et al. collected weak preference judgments from both trained judges and crowdsourcing workers to empirically explore the inter-assessors agreement. Beyond that, the correlation between the collected judgments and the user satisfaction is also investigated. Song et al. introduced an option "same as" in the judging interface and assumed transitivity over the weak preferences in their *Quick-Sort-Judge* method. Additionally, Zhu and Carterette collected weak preferences through a "no preference" option in their research over the user preference for the layout of search results. It seems to us that the strict and weak preferences are regarded as interchangeable in many works. However whether preference judgments with and without ties are the same in terms of judgment quality and judgment efforts remains unclear.

Reduce the number of judgments. The quadratic nature of the number of judgments required is overwhelming in practice (Bashir et al., 2013). Therefore, one important topic regarding preference judgments is to reduce the number of judgments.

Assuming transitivity can dramatically bring down the number of judgments from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$ (Carterette et al., 2008). To utilize transitivity, Rorvig verified the transitivity among judgments from a group of undergraduates. Carterette et al. tested transitivity among judgments from six trained judges, finding that the transitivity holds for 99% of document triples. Moreover, both works applied strict preferences in their empirical studies. Meanwhile, follow-up works tend to extend this property to weak preferences (Song et al., 2011). In this chapter, we attempt to examine the transitivity over both strict and weak preference judgments when collected via crowdsourcing.

Beyond transitivity, several attempts to further bring down the number of judgments were made. Carterette et al. proposed to remove 20% "Bad" judgments by assigning them as worse than others. Niu et al. addressed the expensiveness by only determining a full order for top-k search results, in awareness of the nature that top results are more important to users, reducing the complexity to $\mathcal{O}(N_d \log k)$. Actually, the documents labeled as "Bad" in (Carterette et al., 2008) and the documents out of top-k in (Niu et al., 2012) can be regarded as special cases of tie partitions–a single tie partition with low relevance documents. However, we argue that the reduction of the number of judgments is limited compared with a real tie option, which is especially true for the "Bad" judgments, given that the limited number of documents that are totally off-topic in practice. Moreover, the top-k ground-truth ranking from (Niu et al., 2012) is more suitable for learning to rank algorithms, and may lead to bias for evaluation purpose especially when smaller k is used. A small k leads to a small number of judgments, but results in low recall, treating a lot of relevant or partially relevant documents as the same with non-relevant documents; meanwhile a large k immediately leads to a large number of judgments.

Crowdsourcing for relevance judgments. Existing works examined different ways to collect judgments from crowdsourcing (Grady and Lease, 2010) and provided a proper model to follow in collecting graded judgments from crowdsourcing (Alonso and Baeza-Yates, 2011). Alonso and Mizzaro demonstrated that it is possible to replace graded judgments from TREC using crowdsourcing. Additionally, Kazai et al. compared graded and preference judgments from both trained judges and crowdsourcing, highlighting that preference judgments are especially recommended for crowdsourcing, where judgment quality can be close to the one from trained judges. Different from this work, Kazai et al. measured agreement based on individual judgments, instead of aggregated ones. As mentioned in (Alonso and Mizzaro, 2012), it is the aggregated judgments that can be used in practice. Moreover, the judgment quality is measured in terms of the agreement relative to user clicks, whereas in our work, the measurement is based on judgments from the TREC Web Track. Thereby, in the regards of empirical analysis over judgment quality, our work can be regarded as an extension to both (Alonso and Mizzaro, 2012) and (Kazai et al., 2013).

***Quick-Sort-Judge*.** In our empirical analysis, we employ the labeling mechanism *Quick-Sort-Judge* from (Song et al., 2011), similar to a randomized *QuickSort* method. In *Quick-Sort-Judge*, during each iteration, a document is randomly chosen as a pivot document, denoted as d_p. Thereafter, all remaining documents are grouped into worse than ($\prec d_p$), better than ($\succ d_p$) or tied with ($\sim d_p$) per manual judgments. The mechanism terminates when all documents have been recursively sorted. Note that, within each iteration, the documents on different sides of the pivot document are not manually judged, instead preferences between such document pairs are inferred exploiting transitivity.

4.3 Reducing the Number of Judgments with Ties

In this section, we highlight the role of ties in preference judgments, which have been introduced in existing works, but without noticing their potential in reducing the number of judgments. The ultimate judgment cost is the multiplication of the number of judgments and the cost per judgment, and our discussion mainly focuses on the number of judgments. We assume transitivity among preference judgments which might be over-optimistic in practice. We argue that, however, the collection of transitive judgments, and the design of judgment mechanisms that can tolerate intransitive judgments are orthogonal to this work. Thus, in the following we investigate the number of judgments when allowing for ties analytically and empirically.

4.3.1 Theoretical Analysis

We reexamine the expected number of preference judgments when allowing for ties based on *Quick-Sort-Judge* from (Song et al., 2011) as introduced in Section 4.2.

Notation. Given query q, we denote a set of documents as $\mathcal{D}$, and thus $N_d = |\mathcal{D}|$. Akin to the notation in (Song et al., 2011), in the ground-truth ranking of documents on $\mathcal{D}$, documents that are mutually tied constitute N_t tie partitions, which are denoted as $t_1, t_2, \cdots, t_{N_t}$. Within an individual tie partition t_i, documents are labeled with the same grade or are judged as mutually tied. For example, given the ground-truth ranking of documents as follows.

$$d_5 \sim d_4 \succ d_3 \sim d_2 \succ d_1$$

It can be represented as $t_1 \prec t_2 \prec t_3$, where $t_1 = \{d_1\}$, $t_2 = \{d_2, d_3\}$ and $t_3 = \{d_4, d_5\}$. Given tie partitions $t_i \prec t_j$, we use $\mathcal{D}_{ij}$ to denote documents which lie in between t_i and t_j in the ranking, namely, $\mathcal{D}_{ij} = \{d | t_i \prec d \prec t_j\}$. The set of tie partitions on $\mathcal{D}$ is denoted as $\mathcal{T}$ and $N_t = |\mathcal{T}|$. We introduce $\beta = N_d / N_t$, denoting the average number of documents per tie partition. Thus $\beta = 1$ corresponds to strict preference judgments. Manual judgments can be categorized into two kinds: non-tie judgments, namely $\prec$ and $\succ$, which sort tie partitions; and tie judgments, namely $\sim$, which cluster documents into tie partitions. Correspondingly, the total number of judgments, denoted as N_{jud}, can be split into the number of non-tie judgments, denoted as N_{ntie}, and the number of tie judgments, denoted as N_{tie}. And N_{ntie} can be further made more fine-grained to judgments that determine relative order between a pair of tie partitions t_i and t_j, denoted as N_{ij}, namely, $N_{ntie} = \sum_{t_i, t_j \in \mathcal{T}} N_{ij}$.

Pivot document d_p	$t_i \prec d_p \prec t_j$	$d_p \in t_i$	$d_p \in t_j$
N_{ij}	0	$\lvert t_j\rvert$	$\lvert t_i\rvert$
$P(N_{ij})$	$\frac{\lvert\mathcal{D}_{ij}\rvert}{\lvert t_i\rvert+\lvert\mathcal{D}_{ij}\rvert+\lvert t_j\rvert}$	$\frac{\lvert t_i\rvert}{\lvert t_i\rvert+\lvert\mathcal{D}_{ij}\rvert+\lvert t_j\rvert}$	$\frac{\lvert t_j\rvert}{\lvert t_i\rvert+\lvert\mathcal{D}_{ij}\rvert+\lvert t_j\rvert}$
$E(N_{ij})$		$\frac{2\lvert t_i\rvert\lvert t_j\rvert}{\lvert t_i\rvert+\lvert\mathcal{D}_{ij}\rvert+\lvert t_j\rvert}$	

Table 4.1: The distribution and expectation of N_{ij}, namely, the number of judgments to determine the relative order of two tie partitions t_i and t_j.

Assumptions. As mentioned, our analysis is based on the transitivity assumption. For example, from $d_i \prec d_j$ and $d_j \sim d_k$, we can infer that $d_i \prec d_k$. The transitivity can be applied among tie partitions. For instance, given t_i and t_j, by judging $d_k \in t_i$ and $d_l \in t_j$ as tied, one can get $t_i \sim t_j$ according to transitivity. In addition, we assume that $|t_.| = \beta$, namely, tie partitions have the same size. This actually assumes when two documents are judged as equally relevant if they are at most β positions apart from each other in the ranking of all documents, similar to the analysis from the fourth section in (Radinsky and Ailon, 2011). Note that the size of different tie partitions is more skewed in practice, and this assumption is used to simplify Equation 4.1.

Non-tie judgments: sort the tie partitions. For the non-tie judgments, the number of judgments is analyzed following the analysis for randomized *QuickSort* algorithm (Cormen, 2009). Conceptually, we index these tie partitions according to their ground-truth order, namely, $t_1 \prec t_2 \prec, \cdots, t_i \prec t_j, \cdots, t_{N_t}$. To approach this ground-truth order, one needs to determine the relative order for each pair of tie partitions, say t_i and t_j.

Therefore, one has to either select pivot document d_p from t_i or t_j, resulting in $|t_j|$ or $|t_i|$ judgments respectively, or select a pivot document d_p in between t_i and t_j, namely $d_p \in \mathcal{D}_{ij}$, leading to 0 judgments. In the former case, assuming $d_p \in t_i$, one needs to judge d_p relative to each document in t_j and make $|t_j|$ judgments. In the latter case, the relative order between t_i and t_j is inferred from the judgments between them and d_p, e.g, $t_i \prec d_p, t_j \succ d_p \implies t_i \prec t_j$. The distribution of the random variable N_{ij} is summarized in Table 4.1.

The expected total number of non-tie judgments $E(N_{ntie})$ can be now computed as follows.

$$\begin{aligned} E(N_{ntie}) = E(\sum_{t_i,t_j \in \mathcal{T}} N_{ij}) &= \sum_{i=1}^{N_t-1} \sum_{j=i+1}^{N_t} E(N_{ij}) \\ &= \sum_{i=1}^{N_t-1} \sum_{j=i+1}^{N_t} \frac{2|t_i||t_j|}{|t_i| + |\mathcal{D}_{ij}| + |t_j|} \end{aligned} \tag{4.1}$$

As aforementioned, to further simplify the equation, we assume that tie partitions have equal size, leading to Equation 4.2, where $H_{N_t} = \sum_{k=1}^{N_t} \frac{1}{k}$ is the n_t-th harmonic number, which is in

$\mathcal{O}(\log N_t)$ (Cormen, 2009).

$$\begin{aligned} E(N_{ntie}) &= \sum_{i=1}^{N_t-1} \sum_{j=i+1}^{N_t} \frac{2\beta^2}{\beta(j-i+1)} \\ &= 2\beta \sum_{i=1}^{N_t-1} \sum_{k=2}^{N_t-i+1} \frac{1}{k} \\ &< 2\beta \sum_{i=1}^{N_t} H_{N_t} = 2\beta N_t H_{N_t} \end{aligned} \tag{4.2}$$

Tie judgments: generate tie partitions. When two documents are judged as tied, they are put into the same tie partition. For tie partition t_i, one needs to make $|t_i|$ tie judgments. Therefore, the total number of tie judgments is $E(N_{tie}) = \sum_{i=1}^{N_t} |t_i| = N_d$.

Total number of judgments. Henceforth, the expected total number of judgments equals the sum of the aforementioned two parts as in Equation 4.3, which is in $\mathcal{O}(N_d \log N_t)$.

$$\begin{aligned} E(N_{jud}) &= E(N_{ntie}) + E(N_{tie}) \\ &< 2\beta N_t H_{N_t} + N_d \end{aligned} \tag{4.3}$$

Compared with strict preferences, ties actually produce coarser ground-truth rankings. This can be seen from the analytical results $\mathcal{O}(N_d \log N_t)$ from Section 4.3.1: when $N_t = N_d$ ($\beta = 1$) it becomes strict preferences; and the number is reduced when $N_t < N_d$, where more documents are "squeezed" into a single tie partition. Meanwhile, the ground-truth ranking of documents is simplified to the ranking of tie partitions. In the example from Section 4.3.1, $d_2 \sim d_3$ and $d_4 \sim d_5$ are in the ground-truth ranking, meaning that the ground-truth relative rankings in between d_2 and d_3 and in between d_4 and d_5 are undetermined. In other words, the relative rankings between them are not considered in the evaluation as in (Carterette and Bennett, 2008). Thus, the ties can be regarded as a compromise between the number of judgments and the judgment granularity.

4.3.2 Empirical Analysis

In this section, we empirically examine the number of judgments required in preference judgments to simulate the ground truth from TREC.

Dataset. Our experiments are based on queries from the 2011–2014 TREC Web Track and the corresponding labeled documents (qrel) for adhoc tasks, including 200 queries and 64k graded judgments. The judgments from TREC contain at most six relevance levels: junk pages (*Junk*), non-relevance (*NRel*), relevance (*Rel*), highly relevant (*HRel*), key pages (*Key*) and navigational pages (*Nav*), corresponding to six graded levels, i.e., -2, 0, 1, 2, 3, 4. The concrete assignments varied from year to year, where *Junk* and *NRel* are always merged as *NRel* in this work, given the limited occurrences of *Junk* judgments (less than 5%). To employ the system rankings from rivaling systems as features in *ActiveSVM* (Tong and Koller, 2001), we also obtained the runs submitted by participants of the TREC Web Track. There are 62 runs from 2011, 48 runs from 2012, 61 runs from 2013, and 42 runs from 2014.

Collecting preference judgments. Ideally, we should rejudge the documents with preference judgments and compare them with the original graded judgments from TREC. However, this is unaffordable given the huge number of document pairs to judge. Thus, we employ the existing graded judgments from the TREC Web Track to create preference judgments in a straightforward manner as in (Cao et al., 2006). In particular, the preference judgments are created for two documents according to the comparison of their graded judgments, namely, if the label for d_1 is l_1 and the label for d_2 is l_2, the preference between these two documents is $d_1 \succ d_2$ when $l_1 > l_2$; $d_1 \sim d_2$ when $l_1 = l_2$ and $d_1 \prec d_2$ otherwise. In this way, after collecting judgments for all document pairs, we can simulate the same ground truth from graded judgments. Note that, in practice, it is unlikely to create exactly the same ground truth from judgments collected with different methods. This setting is mainly for comparing preference and graded judgments under the same condition, and also for guaranteeing that the same amount of ranking information is collected by different competing mechanisms.

Methods under comparison. We compare the number of judgments from three methods: graded judgments, weak preference judgments, and strict preference judgments. The number of judgments in graded judgments simply equals the number of documents. The preference judgments are simulated by randomly selecting document pairs with the established *Quick-Sort-Judge* (Song et al., 2011) as introduced in Section 4.2. Thereafter, in preference judgments with ties, the judgments are simulated by comparing the ground-truth labels of two documents from TREC. For strict preference judgments, given that ties are not allowed, the relative order between documents with the same labels from TREC are further determined by their string identifiers, which are unique and fixed among random experiments. We report the average number of judgments from 1000 repetitions of *Quick-Sort-Judge* for both kinds of preference judgments.

Results. The results are summarized in Figure 4.1. It can be seen that, the judgments from strict preferences are far more than the one when allowing for ties, namely, on average 500% more judgments are required. Compared with the number of judgments required by graded judgments, the numbers are 43% and 773% higher respectively when allowing and not allowing for ties.

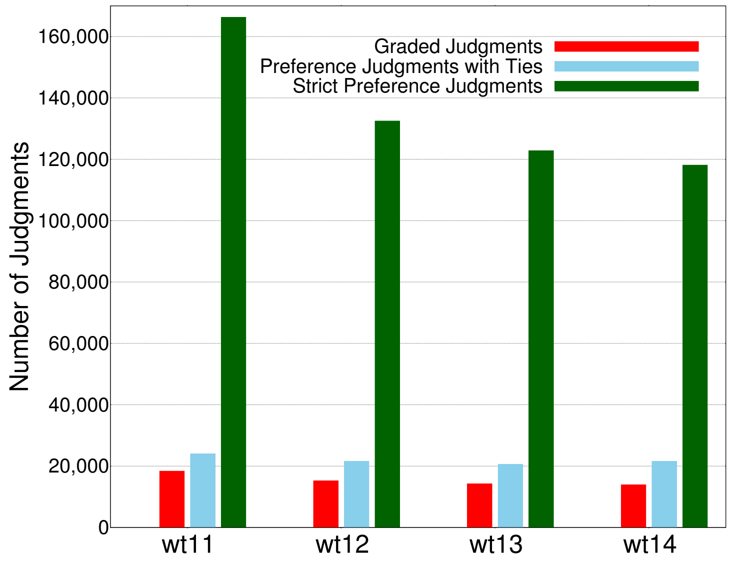


Figure 4.1: The average number of judgments required by graded judgments and by preference judgments with/without ties on TREC Web Track. The x-axis is different years and y-axis represents the number of judgments. The averaged number of judgments from 1000 repetitions is reported as the actual number of judgments for both kinds of preference judgments.

Encode Cluster Effects

In this part, we discuss whether there is potential to reduce the number of judgments with ties beyond *Quick-Sort-Judge*. Similar to the strategy employed in (Wang et al., 2013), ideally, one can first make tie judgments to cluster documents, and thereafter make non-tie judgments to sort the tie partitions. By doing this, the number of tie judgments remains the same, namely N_d. Whereas for non-tie judgments, the number of judgments under $d_p \in t_i$ and $d_p \in t_j$ becomes 1 in Table 4.1, which means that one only needs to judge a pair of documents to determine the relative order of two established tie partitions. Accordingly, the number of judgments is reduced to

$$E(N_{jud}) = 2 \sum_{i=1}^{N_t-1} \sum_{k=2}^{N_t-i+1} \frac{1}{k} + N_d < 2N_t H_{N_t} + N_d,$$

which is in $\mathcal{O}(2N_t \log N_t + N_d)$ and is close to linear when $N_t \ll N_d$.

Inspired by this, we propose a union-find method as labeling mechanism, named *Merge-Tie-Judge*, to make use of the "cluster effect" of ties, by dynamically merging documents into tie partitions in the judgment procedure. Specifically, in the *Merge-Tie-Judge* mechanism, the "cluster effect" is leveraged explicitly. Intuitively, a cluster of documents grows via incoming tie judgments, by either adding a tied document to it or by merging it with other clusters. The follow-up judgments are made based on these merged clusters, the number of which keeps decreasing as more and more tie judgments are made.

```
Input All documents $\mathcal{D}$, document pairs $\mathcal{E}$, initialization of tie probabilities $\mathcal{P}^t$, hyper parameter JudNumber4SVM.
Output Sorted clusters of tied documents: $\mathcal{J}ud = \{jud_{ij} = c_i \succ c_j \text{ or } c_i \prec c_j | c_i, c_j \in \mathcal{C}\}$, where $c_{\cdot} = \{d_i | d_i \in \mathcal{D} \text{ that are tied}\}$.
/* $\mathcal{C}_{\succ c}$ and $\mathcal{C}_{\prec c}$: sets of clusters that are better or worse
   than $c$, initialized as empty.                                   */
Initialization Clusters $\mathcal{C} = \{c_i = \{d_i\} | d_i \in \mathcal{D}\}$, TranTracker $= \{(c_i, \mathcal{C}_{\prec c}, \mathcal{C}_{\succ c}) | c_i \in \mathcal{C}\}$;
StopCondition $\forall\, c_i, c_j \in \mathcal{C}$, $c_i \prec c_j$ or $c_i \succ c_j \in \mathcal{J}ud$
while not StopCondition do
    /* Select two clusters for judgment.                             */
    if $|\mathcal{J}ud| <$ JudNumber4SVM then
        $c_i, c_j, \mathcal{P}^t = ActiveSVM(\mathcal{C}, \mathcal{J}ud)$;
    else
        $c_i, c_j = SelectClusterPair4Judgment(\mathcal{C}, \mathcal{P}^t)$;
    end
    $jud_{ij} = ManualJudgment(c_i, c_j)$;

    if $jud_{ij}$ is $c_i \sim c_j$ then
        /* Merge $c_i$, $c_j$ into new cluster $c_n$, and update $\mathcal{C}$.   */
        $\mathcal{C}$, $c_n$=MergeClusters($c_i$, $c_j$, $\mathcal{C}$)
    end
    /* Update $\mathcal{J}ud$, adding incoming and inferred judgments.
       */
    $\mathcal{J}ud$=UpdateTranTracker(TranTracker, $\mathcal{J}ud$, $jud_{ij}$)

    /* Update $\mathcal{P}^t$ for document pairs that are judged, 0 for
       non-tie and 1 for tie.                                        */
    $\mathcal{P}^t$=UpdateTieProbability($\mathcal{P}^t$, $\mathcal{J}ud$)
end
```

Algorithm 1: *Merge-Tie-Judge*

The basic data structure here is the cluster of tied documents, which is initialized with a single document. When collecting judgments, two clusters are merged when at least one document pair from either side is judged as a tie; meanwhile the transitivity relationship for an individual cluster c is tracked with *TranTracker*, recording clusters that are judged better than ($\mathcal{C}_{\succ c}$) or worse than it ($\mathcal{C}_{\prec c}$). When a document pair is judged as a non-tie ($\prec$ or $\succ$), the transitivity property is applied over all clusters involved, and the *TranTracker* is updated accordingly. For example, if a new judgment indicates $c_i \succ c_j$, then we need to update *TranTracker* for c_i and c_j, as well as for clusters that are better than c_i, i.e., $c \in \mathcal{C}_{\succ c_i}$, and that are worse than c_j, i.e., $c \in \mathcal{C}_{\prec c_j}$, adding $jud_{ij} = c_i \succ c_j$ as well as the inferred pairwise judgments to $\mathcal{J}ud$, namely, $\{jud_{kl} = c_k \succ c_l | c_k \in \mathcal{C}_{\succ c_i}, c_l \in \mathcal{C}_{\prec c_j}\}$.

Another important data structure is $\mathcal{P}^t$, which tracks an estimate of the probability of being tied for every pair of clusters, and $\mathcal{P}^t_{ij} = P(c_i \sim c_j)$. During iterations, the next cluster pair to judge is selected according to $\mathcal{P}^t$: the cluster pair with the largest tie probability is chosen for judgment. It could be initialized randomly or based on prior knowledge, which is introduced later. After initialization, we keep updating $\mathcal{P}^t$ to reflect new judgments, and to compute tie probabilities among emerging clusters. In *UpdateTieProbability*, the tie probability between two emerging clusters is computed as the sum of the tie probability between old cluster pairs residing in either side. This aggregation represents the union of the involved document pairs that are tied. Thus, the larger the clusters, the more likely they are picked out for judgment.

ActiveSVM for tie inference. As mentioned, the tie probability $\mathcal{P}^t$ introduced above can be initialized randomly. We argue that manual judgment procedure is special in the sense that it is too expensive to repeat dozens of times in practice. And the average number of judgments could be misleading, especially when the variance is large. In other words, though the average number of judgments is acceptable, the judgment mechanism may still risk an extremely large number of judgments in practice. Therefore, we propose to remove this randomness by introducing a predictor, i.e., *ActiveSVM*$(\mathcal{C}, \mathcal{J}ud)$ in Algorithm 1. The prediction function will be triggered when positive *JudNumber4SVM* is set. The prediction can be cast as a supervised binary classification problem.

Active Support Vector Machine. The support vector machine (SVM) is employed to make predictions. A document pair d_i, d_j is denoted as p_{ij}, meanwhile its corresponding label is $y_{ij} = -1$ or 1, corresponding to non-tie and tie. The classification problem aims at learning the function between the feature vector of each pair, i.e., $\Phi(p_{ij})$, and the binary label. Given that the ultimate target is to collect labels over document pairs with fewer manual judgments, the number of judgments for making prediction is desired to be small. Henceforth, the *ActiveSVM* with **Ratio Margin** strategy proposed by Tong and Koller is used, which is designed to approach the optimized hyperplane with a small number of labeled data points. In each iteration, for each unlabeled data point, two new classifiers w_+ and w_- are trained by hypothetically assigning the data point a positive or a negative judgment respectively. The next point to label is selected according to the ratio between m_+ and m_-, by picking out $min(\frac{m_+}{m_-}, \frac{m_-}{m_+})$, where $m_.$ is the sum of margins of the classifier defined as follows, where the $\Phi(.)$ represents the a feature vector for a given pair of documents.

$$m_+ = \sum_{p_{ij} \in \text{Support Vectors}} w_+ \Phi(p_{ij})$$
$$m_- = \sum_{p_{ij} \in \text{Support Vectors}} w_- \Phi(p_{ij})$$

Comparison of the number of judgments. Finally, we describe the experiments to examine the proposed *Merge-Tie-Judge* mechanism by comparing it with *Quick-Sort-Judge* (Song et al., 2011). The dataset described in Section 4.3.2 is used. To empirically compare the number of judgments from graded judgments and from preference judgments, we equivalently answer a question: "to approach the same ground truth generated with the graded judgments from TREC, how many judgments are required with preference judgments". We denote this number as *equivalent judgment number* (EQUJN). As discussed above, we further examine the robustness of the proposed mechanism.

Competing mechanisms. The number of judgments with graded judgments is simply the total number of documents to judge, denoted as #Document. As a competitor, we implement *Quick-Sort-Judge* (*QSJ*) from (Song et al., 2011) as introduced in Section 4.2.

We examine the proposed *Merge-Tie-Judge* (*MTJ*), where ActiveSVM is used to initialize $\mathcal{P}^t$. We employ rankings from different systems in TREC as features to train ActiveSVM. In our pilot experiments, the optimal setting of *JudNumber4SVM* varies a lot over different queries, as results of the difference of the quality of system rankings (features) and of the instinct difficulty to make prediction etc.. Therefore, we regard the different settings of *JudNumber4SVM* as a random factor, and demonstrate that the number of judgments over one year is robust within a range of settings for *JudNumber4SVM*. In particular, we investigate when *JudNumber4SVM* $\in [5, 35]$. In addition, we also include the variants of *Merge-Tie-Judge* when initialized $\mathcal{P}^t$ randomly, and denote it as *Merge-Tie-Judge-Random* (*MTJR*). The results for *Merge-Tie-Judge-Random* are also based on 300 repeats.

Equivalent number of judgments. In this section, we examine the *equivalent judgment number* from different competing mechanisms. The results for *Quick-Sort-Judge* are summarized in Table 4.2. The results for *Merge-Tie-Judge* and its comparison relative to *Quick-Sort-Judge* are summarized in Table 4.3. For *Merge-Tie-Judge*, we report the results under the average, best, and worst situations with different *JudNumber4SVM* in Table 4.3. We can see that the number of judgments is reduced significantly by 5.9%, 6.8% and 4.7% under average, best, and worst situations respectively, in comparison with the average number of judgments in *Quick-Sort-Judge*. Beyond that, the results from MERGE-TIE-JUDGE, together with MERGE-TIE-JUDGE-RANDOM

YEAR	#Document	*QSJ*	% more	CV
WT11	19,381	25,122	29.6%	0.159
WT12	16,055	22,587	40.7%	0.176
WT13	14,474	20,897	44.4%	0.167
WT14	14,429	22,331	54.8%	0.154
SUM	64,339	90,937	41.3%	-

Table 4.2: *Equivalent judgment number* required by *Quick-Sort-Judge* (*QSJ*). The absolute number of judgments (EQUJN), the relative comparison with document number (% more), and the coefficient of variance (CV) are reported.

YEAR	Average EQUJN	% more	Best EQUJN	% more	Worst EQUJN	% more	CV
WT11	23,818 (5.2%)	22.9%	23,680 (5.7%)	22.2%	23,977 (4.6%)	23.7%	0.0027
WT12	21,087 (6.7%)	31.3%	20,845 (7.7%)	29.8%	21,289 (5.8%)	32.6%	0.0046
WT13	19,557 (6.4%)	35.1%	19,365 (7.3%)	33.8%	19,887 (4.8%)	37.4%	0.0055
WT14	21,106 (5.5%)	46.3%	20,887 (6.5%)	44.8%	21,546 (3.5%)	49.3%	0.0083
SUM	85,568 (5.9%)	33.0%	84,777 (6.8%)	31.8%	86,699 (4.7%)	34.8%	-

Table 4.3: *Equivalent judgment number* when using *Merge-Tie-Judge* with *ActiveSVM*. The statistics reported are based on $JudNumber4SVM \in [5, 35]$. Both absolute number of judgment (EQUJN) and the relative comparison with document number (% more) are reported for average, best (minimum number of judgments) and worst (maximum number of judgments) situations. The number in the bracket is the relative reduction w.r.t. mean value in *Quick-Sort-Judge*. The coefficient of variation (CV) is reported in the rightmost column.

and *Quick-Sort-Judge* are visualized in Figure 4.2, where all randomized mechanisms are reported with box-and-whisker plots, encoding statistics of mean, minimum, maximum, and the 95% confidence interval. We can see that there is no overlap of the 95% confidence interval from two variates of *Merge-Tie-Judge* and the one from *Quick-Sort-Judge*. Therefore, we can conclude that in terms of the number of judgments, both *Merge-Tie-Judge* and *Merge-Tie-Judge-Random* are significantly better than *Quick-Sort-Judge*.

Robustness. As mentioned, the robustness of the judgment mechanisms is also important. Especially, one may desire even under the worst situation the number of judgments from a mechanism still be close to the number of judgments on average. From Figure 4.2, *Quick-Sort-Judge* fails to meet this expectation since the largest number of judgments could be multiple times larger. We report the coefficient of variation (CV) to quantify the robustness in Table 4.2 and 4.3, which equals the ratio of the standard deviation to the mean. We can see that *Merge-Tie-Judge* is much more robust among a wide range of *JudNumber4SVM*, and the coefficient of variance is only

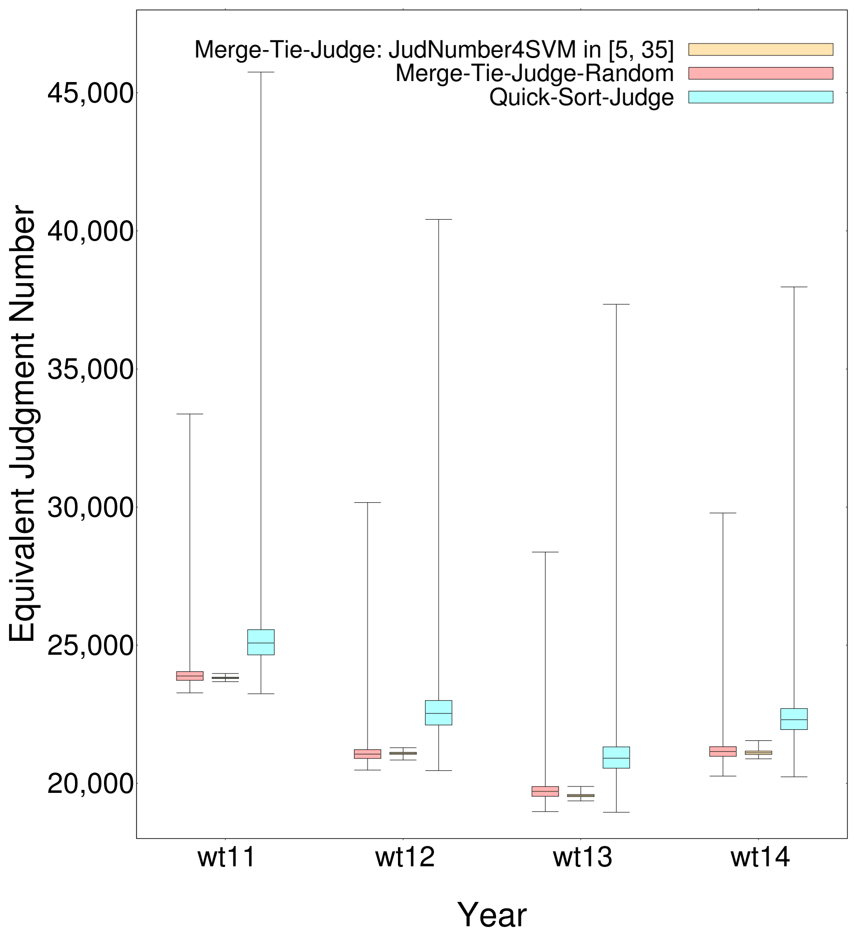
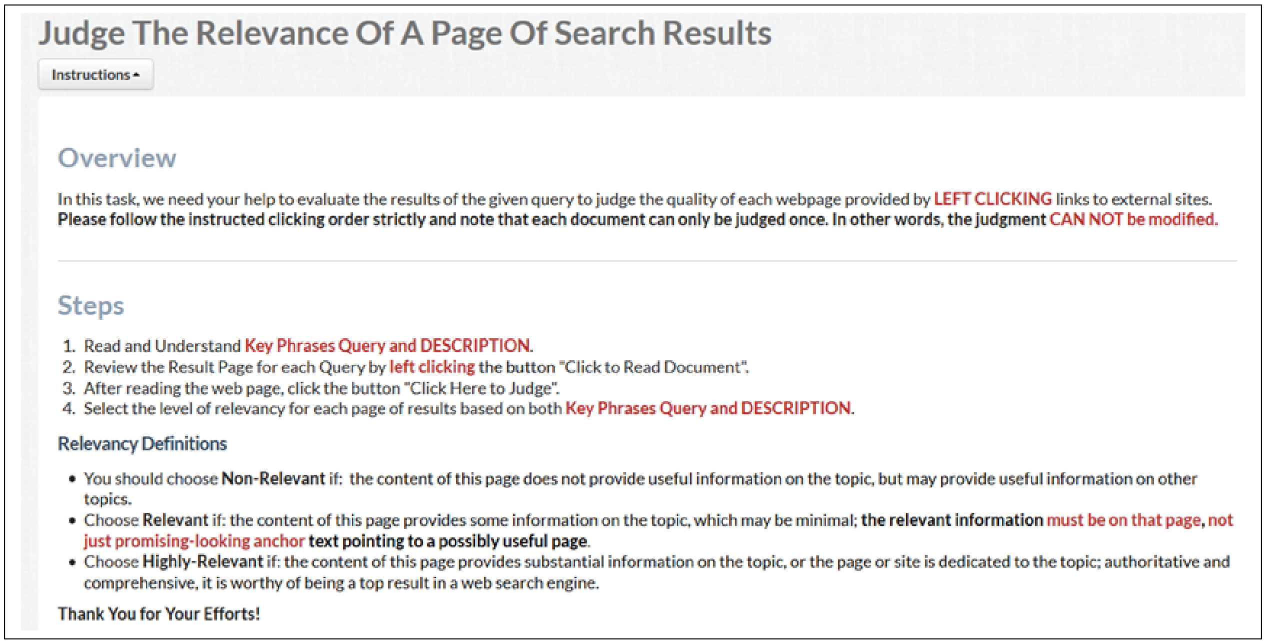


Figure 4.2: Comparison of number of judgments from different mechanisms. The x-axis is different years and y-axis represents the *equivalent judgment number*. The fewer *equivalent judgment number* the better. The randomized *Quick-Sort-Judge* is reported as baselines; the *Merge-Tie-Judge*, which implements *ActiveSVM*, and *Merge-Tie-Judge-Random* are both reported. For random mechanism, the mean, minimum, maximum as well as the 95% confidence interval are plotted.

3.2% of the one from *Quick-Sort-Judge*. In addition, in Figure 4.2, from the distance between minimum and maximum number of judgments, as well as from the length of their confidence interval, it is obvious that both *Merge-Tie-Judge* variants are much more robust. Finally, compared with *Merge-Tie-Judge-Random*, *Merge-Tie-Judge* enjoy a better robustness by utilizing *ActiveSVM*, reducing the possible number of judgments into a rather small range.


Judge The Relevance Of A Page Of Search Results

Instructions

Overview

In this task, we need your help to evaluate the results of the given query to judge the quality of each webpage provided by LEFT CLICKING links to external sites. **Please follow the instructed clicking order strictly and note that each document can only be judged once. In other words, the judgment CAN NOT be modified.**

Steps

1. Read and Understand Key Phrases Query and DESCRIPTION.
2. Review the Result Page for each Query by left clicking the button "Click to Read Document".
3. After reading the web page, click the button "Click Here to Judge".
4. Select the level of relevancy for each page of results based on both Key Phrases Query and DESCRIPTION.

Relevancy Definitions

- You should choose **Non-Relevant** if: the content of this page does not provide useful information on the topic, but may provide useful information on other topics.
- Choose **Relevant** if: the content of this page provides some information on the topic, which may be minimal; **the relevant information must be on that page, not just promising-looking anchor text pointing to a possibly useful page.**
- Choose **Highly-Relevant** if: the content of this page provides substantial information on the topic, or the page or site is dedicated to the topic; authoritative and comprehensive, it is worthy of being a top result in a web search engine.

Thank You for Your Efforts!

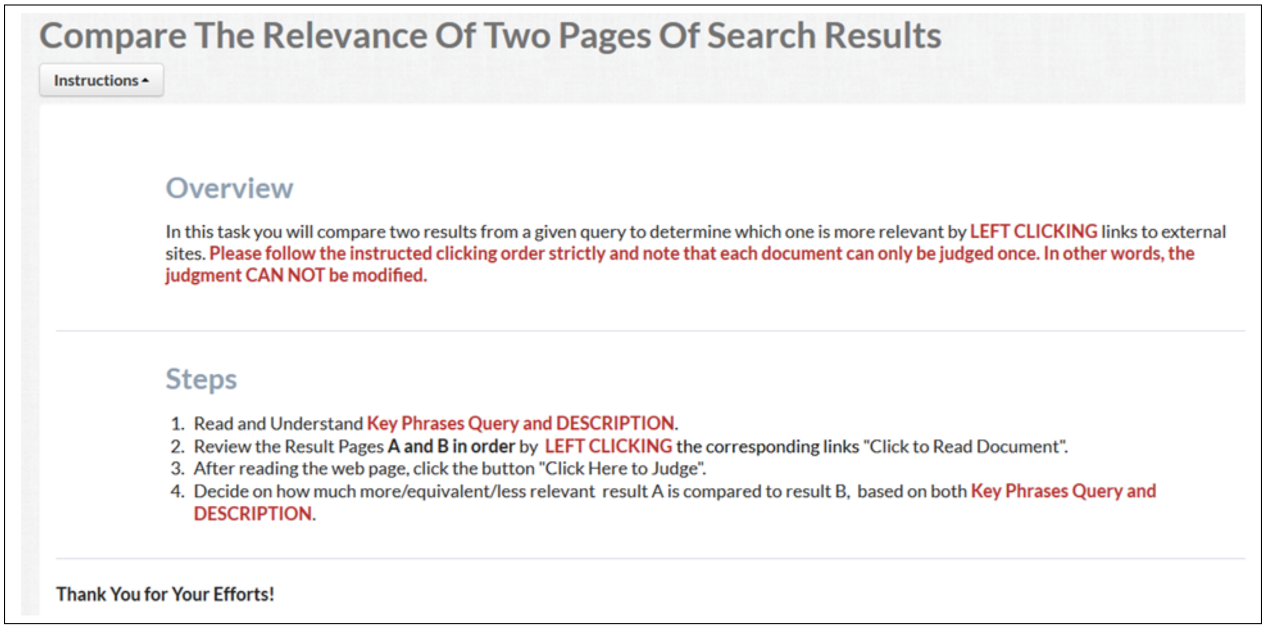
Figure 4.3: The instructions used to collect graded judgments.

4.4 Crowdsourcing Task and General Statistics

To answer **RQ2–RQ4** from Section 4.1, we empirically compare different kinds of preference judgments relative to graded judgments via CrowdFlower. Using topics and pooled documents from the TREC Web Track we collect graded judgments, weak preference judgments, and strict preference judgments. In this section, we introduce our configuration for the crowdsourcing task, and the setup for **RQ2–RQ4** respectively. The task instructions for guiding the assessors to judge graded and preference judgments are shown in Figures 4.3 and 4.4 respectively.

4.4.1 Task Configuration

User interface. We display queries together with their description from the TREC Web Track 2013 & 2014. Judges are instructed to consider both the query and its corresponding description as in Figure 2.2. To help them understanding the topic, we also display a link to run the query against a commercial web search engine. When collecting preference judgments, we show the query and description together with two documents (A and B) and ask judges "Which document is more relevant to the query?". When collecting strict preferences, judges can choose between the options "Document A is more relevant" and "Document B is more relevant". A third option "Document A and B are equivalent" is added, when collecting weak preferences. When collecting graded judgments, the query and description are shown together with a single document. Judges are asked "How well does the document match the query?" and can click on one of the grades


Compare The Relevance Of Two Pages Of Search Results

Instructions

Overview

In this task you will compare two results from a given query to determine which one is more relevant by LEFT CLICKING links to external sites. Please follow the instructed clicking order strictly and note that each document can only be judged once. In other words, the judgment CAN NOT be modified.

Steps

1. Read and Understand Key Phrases Query and DESCRIPTION.
2. Review the Result Pages **A and B in order** by LEFT CLICKING the corresponding links "Click to Read Document".
3. After reading the web page, click the button "Click Here to Judge".
4. Decide on how much more/equivalent/less relevant result A is compared to result B, based on both Key Phrases Query and DESCRIPTION.

Thank You for Your Efforts!

Figure 4.4: The instructions used to collect preference judgments.

"Highly Relevant", "Relevant", and "Non-Relevant". In our instructions we include the same definitions of grades from TREC. The judgment interfaces for graded and preference judgments are displayed in Figure 4.5 and 4.6.

Quality control. Unique tasks, in our case judgments, are referred as rows in CrowdFlower. Multiple rows are grouped into a page, which is the basic unit for payment and quality control. The major means to control quality are test questions, that is, rows with a known expected input from workers. Test questions can be used to run a qualification quiz, which workers have to complete upfront. By thresholding on their accuracy in the qualification quiz, unreliable workers can be filtered out. Moreover, test questions can be interspersed with rows to continuously control the quality of work. Workers can thus be banned once their accuracy on interspersed test questions drops below a threshold. The accuracy threshold is set as 0.7, following the default on CrowdFlower.

Job settings. When collecting graded judgments a page consists of eleven judgments and a test question, and workers are paid \$0.10 on each successful completion. When collecting preference judgments, we pack eight document pairs and a test question into each page, and pay workers \$0.15 on successful completion. The rationale behind the different pays is that workers receive the same amount of \$0.0083 per document read. Each row is shown to workers until three trusted judgments have been collected.

Selection of queries and documents. Queries and documents are sampled from the TREC Web Track 2013 & 2014. From the 100 available queries, we sample a subset of twelve queries.[3]

[3]Queries are available in `http://trec.nist.gov/data/webmain.html`.

Among the sampled queries, one query is marked as ambiguous by TREC, five queries are marked as unambiguous (single), and six queries are faceted. The original relevance judgments contain up to six relevance levels: junk pages (*Junk*), non-relevant (*NRel*), relevant (*Rel*), highly relevant (*HRel*), key pages (*Key*), and navigational pages (*Nav*), corresponding to six graded levels, i.e., -2, 0, 1, 2, 3, 4. Different from other grades, *Nav* indicates a document can satisfy a navigational user intend, making the comparison relative to other documents depend on the information intent from the crowdsourcing judges. Hence, in our work, documents labeled *Nav* together with documents labeled *Junk* are removed. Due to their limited occurrences, documents labeled *Key* and *HRel* are both regarded as highly relevant. For each query we determine two sets of documents. Each set consists of twelve documents selected uniformly across graded levels, resulting in four documents per graded level. The first set is used to collect judgments; the second set serves to create test questions. When collecting graded judgments, the selected documents are directly used. To collect preference judgments, we generate for each query all 66 pairs of documents and randomly permute each document pair. Test questions are generated treating the judgments from TREC as ground truth. To ensure that workers on CrowdFlower see the same documents as trained judges from TREC, we host copies of ClueWeb12[4] documents on our own web server.

Time consumption. To monitor the time consumed for reading documents and making judgments, we proceed as follows. We record the timestamp when judges start reading the shown document(s). To display available options for judging, workers have to click on a button "Click here to judge", and we record the instant when this happens. As a last timestamp, we record when the worker selects the submitted option. In recording timestamps, the order of clicks from judges are restricted by customized JavaScript, e.g., "Click here to judge" button is enabled only after document(s) is (are) read. We thus end up with three timestamps, allowing us to estimate the reading time, as the time passed between the first two timestamps, and the judgment time, as the time passed between the last two.

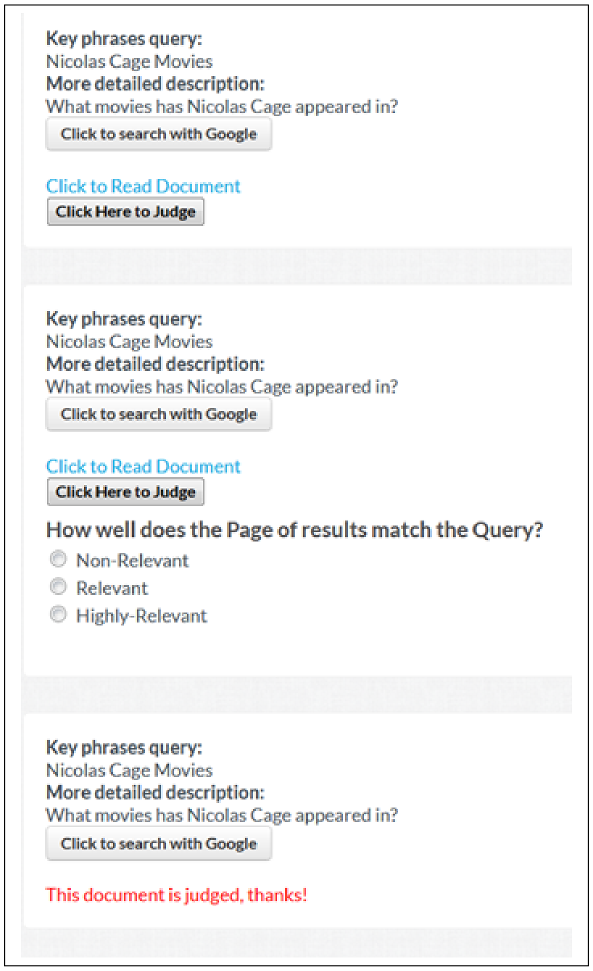
Judgment aggregation. As mentioned, at least three trusted judgments are collected for each row. One straightforward option to aggregate them is to use majority voting as suggested in (Alonso and Baeza-Yates, 2011). However, in our setting, a simple majority vote may not break ties, given that there are more than two options to choose from. As a remedy we use workers' accuracies, as measured on test questions, in a weighted majority voting to break ties.

4.4.2 General Statistics

Table 4.4 summarizes general statistics about the collected judgments. The collected judgments are publicly available.[5]

[4] http://lemurproject.org/clueweb12/index.php

[5] http://people.mpi-inf.mpg.de/~khui/data/ecir17empirical

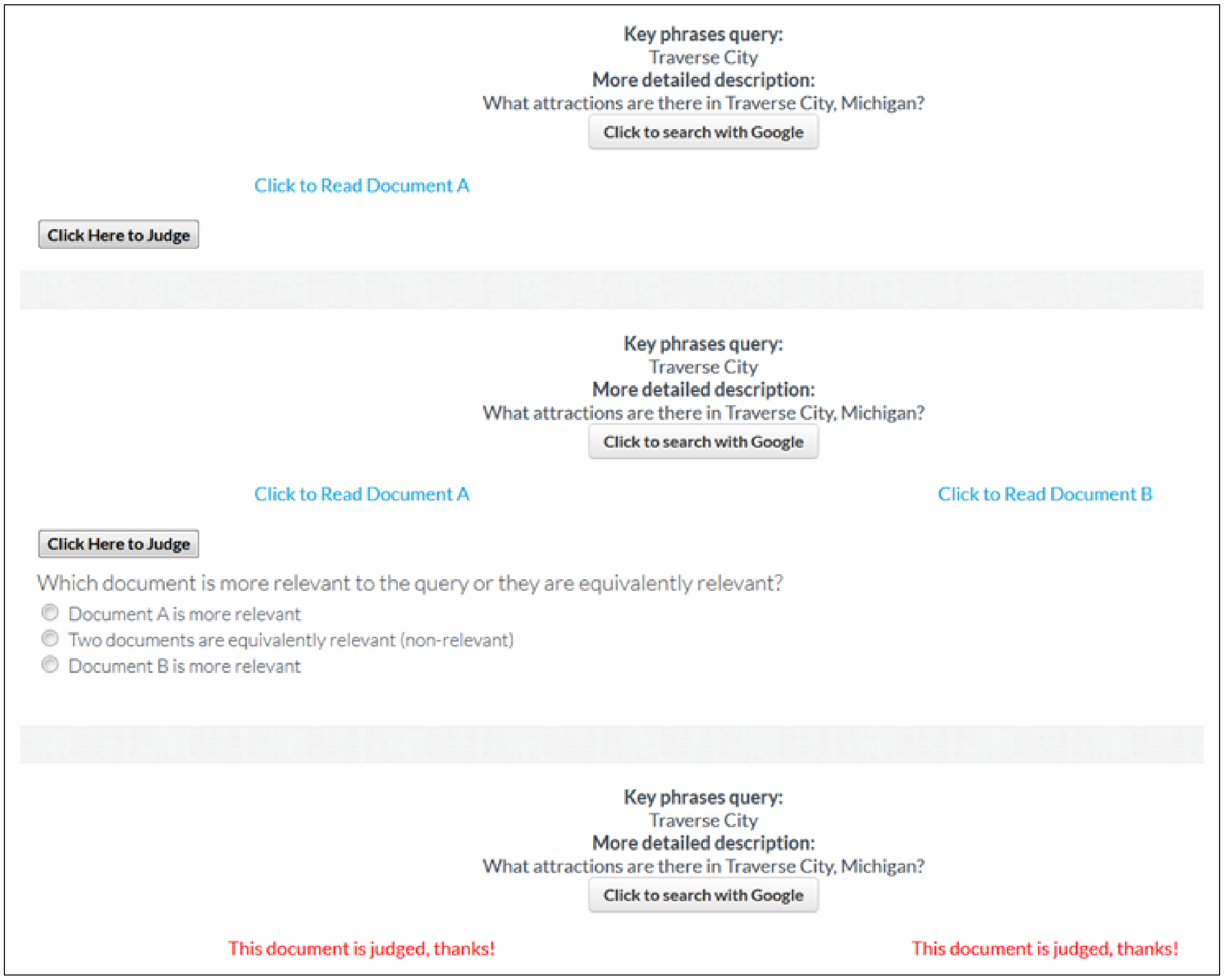

Key phrases query:
Nicolas Cage Movies
More detailed description:
What movies has Nicolas Cage appeared in?
Click to search with Google
Click to Read Document
Click Here to Judge

Key phrases query:
Nicolas Cage Movies
More detailed description:
What movies has Nicolas Cage appeared in?
Click to search with Google
Click to Read Document
Click Here to Judge
How well does the Page of results match the Query?
- Non-Relevant
- Relevant
- Highly-Relevant

Key phrases query:
Nicolas Cage Movies
More detailed description:
What movies has Nicolas Cage appeared in?
Click to search with Google
This document is judged, thanks!

Figure 4.5: The judgment interface for graded judgments. Three judgment units are displayed. The upper one shows before judgments; the middle one demonstrates when user clicks the button "Click Here to Judge" after reading the document; and the bottom one is displayed after user makes judgments by clicking one of the three options.

Inter-judge agreement. Similar to (Alonso and Mizzaro, 2012), Fleiss' κ is computed over each query and average Fleiss' κ among all queries is reported in Table 4.4. To put our results in context, we merge "Highly-Relevant" with "Relevant" and convert graded to binary judgments, ending up with Fleiss' $\kappa = 0.269$, which is close to 0.195 reported in (Alonso and Mizzaro, 2012). One reason for that might be we only collect three judgments for each query document pair, whereas five judgments were collected in (Alonso and Mizzaro, 2012). In addition, Kazai et al. reported Fleiss' $\kappa = 0.24$ (cf. Table 2 PC (e) therein) among weak preference judgments from crowdsourcing, which approximates 0.253 in our work. We further conduct two-tailed Student's t-tests between the three kinds of judgments over different queries. The p-value between strict preferences and graded judgments is smaller than 0.001; between weak preferences and graded judgments is 0.314; whereas it is 0.005 between the two kinds of preference judgments. It can be seen that the judges achieve better inter-agreement for strict preferences than for the others,


Key phrases query:
Traverse City
More detailed description:
What attractions are there in Traverse City, Michigan?

Key phrases query:
Traverse City
More detailed description:
What attractions are there in Traverse City, Michigan?

Which document is more relevant to the query or they are equivalently relevant?
- Document A is more relevant
- Two documents are equivalently relevant (non-relevant)
- Document B is more relevant

Key phrases query:
Traverse City
More detailed description:
What attractions are there in Traverse City, Michigan?

This document is judged, thanks!
This document is judged, thanks!

Figure 4.6: The judgment interface for preference judgments. Three judgment units are displayed. The upper one is before judgments; the middle one demonstrates when user clicks the button “Click Here to Judge” after reading both documents; and the bottom one displays after user makes judgments by clicking one of the three options.

meanwhile there is no significant difference between weak preferences and graded judgments. This aligns with the observations from (Carterette et al., 2008), that strict preferences exhibit higher inter-judge agreement. The introduction of “ties” reduces the inter-judge agreement, which might be due to more options being available.

4.5 Judgment Quality via Crowdsourcing

To answer the **RQ2** from Section 4.1, we compare the quality of three kinds of judgment collected via crowdsourcing in terms of their agreement with judgments from TREC (qrel). We employ both percentage agreement, which counts the agreed judgments and divides it by the number

	Graded Jud		Strict Pref		Weak Pref	
Total Cost	$9.36		$62.10		$76.80	
#Judgments	919		2,760		2,931	
# Per Judge	28.80		55.00		20.10	
Fleiss' κ	0.170		0.498		0.253	
Distribution of Judgments						
"Highly-Relevant"	28%		$A \succ B$	51%	$A \succ B$	30%
"Relevant"	43%		$A \prec B$	49%	$A \prec B$	31%
"Non-Relevant"	29%		-		$A \sim B$	39%

Table 4.4: General statistics about judgments collected using crowdsourcing.

TREC	*Highly-Relevant*	*Relevant*	*Non-Relevant*	#Total
Highly-Relevant	47.9%	37.5%	14.6%	48
Relevant	31.2%	54.2%	14.6%	48
Non-Relevant	4.1%	39.6%	56.3%	48

Table 4.5: Agreement between graded judgments from crowdsourcing (columns) and TREC (rows).

TREC	$A \prec B$	$A \succ B$	#Total
$A \prec B$	83.0%	17.0%	282
$A \sim B$	46.8%	53.2%	216
$A \succ B$	20.4%	79.6%	294

Table 4.6: Agreement between preferences judgments from crowdsourcing (columns) and the one inferred from TREC judgments (rows) for strict preference judgments.

of total judgments, and Cohen's κ as in (Alonso and Mizzaro, 2012), and use the latter for two-tailed Student's t-tests. When evaluating preference judgments from crowdsourcing, judgments from TREC are first converted to preference judgments, by comparing labels over two documents, resulting in "better than", "worse than", or "tie". The percentage agreement over graded judgments to judgments from TREC are summarized in Table 4.5. Results for the two kinds of preference judgments are summarized in Table 4.6 and 4.7. In all these tables, the percentage is normalized per row.

TREC	$A \prec B$	$A \sim B$	$A \succ B$	#Total
$A \prec B$	62.8%	30.9%	6.3%	285
$A \sim B$	17.6%	59.7%	22.7%	216
$A \succ B$	7.6%	32.0%	60.5%	291

Table 4.7: Agreement between preferences judgments from crowdsourcing (columns) and the one inferred from TREC judgments (rows) for weak preference judgments.

To put our results in context, we first measure agreement based on binary judgments, by merging the grades *Relevant* and *Highly-Relevant* in both TREC judgments and graded judgments from crowdsourcing. In (Alonso and Mizzaro, 2012), percentage agreement equals 77% and Cohen's $\kappa = 0.478$, relative to judgments from TREC-7 and TREC-8. Meanwhile we obtain 75.7% and Cohen's $\kappa = 0.430$ – slightly lower values. We argue that this is due to the document collections in use: ClueWeb12, used in our work, consists of web pages which are more diverse and noisy, making it harder to judge; whereas disk 4&5 used in TREC-7 and TREC-8 consist of cleaner articles[6]. When using three grades, graded judgments from crowdsourcing achieve 52.8% and Cohen's $\kappa = 0.292$ relative to judgments from TREC. The percentage agreement is 59.1% and Cohen's $\kappa = 0.358$ for strict preferences, whereas for weak preferences the numbers are 61% and 0.419 respectively. Compared with graded judgments from crowdsourcing, the corresponding p-values from paired sample t-tests over Cohen's κ among queries are 0.259 and 0.052, indicating weak preference judgments agree with TREC judgments better.

Note that, however, for documents with the same grade in TREC a tie is inferred, whereas strict preferences do not permit tie judgments. From Table 4.6, it can be seen that 216 document pairs are inferred as tied, where agreement is zero for strict preferences currently. To mitigate this mismatch, in line with (Carterette et al., 2008), tie judgments in inferred preference judgments are redistributed as "A is better" or "B is better". In this redistribution, an agreement is assumed, coined as aar. In other words, the 216 document pairs that are inferred as tied in Table 4.6 are redistributed so that $216 \times aar$ random pairs are assigned with the same judgments as in collected strict preference judgments. The logic behind this is that the ground-truth strict preferences over these inferred ties are unknown and we need to assume an agreement over them to make strict preference judgments comparable. Thereby, two groups of agreement are reported for strict preference judgments at assumed agreement rates $aar = 50\%$ and 80%, respectively corresponding to random agreement and the average agreement under non-tie situations (average of 83% and 79.6% in Table 4.6). Without influencing comparison results, graded judgments from crowdsourcing are also converted to the form of preference judgments. In Table 4.8, it can be seen that Cohen's $\kappa = 0.530$ for strict preferences when assuming $aar = 50\%$, and the value for weak

[6] http://trec.nist.gov/data/docs_eng.html

Query	Strict Preferences				Weak Preferences		Graded Judgments	
	break tie $aar = 50\%$		break tie $aar = 80\%$					
	agreement	Cohen's κ	agreement	Cohen's κ	agreement	Cohen's κ	agreement	Cohen's κ
216	77%	0.594	85%	0.710	65%	0.466	53%	0.269
222	76%	0.569	83%	0.680	59%	0.391	65%	0.474
226	77%	0.589	79%	0.611	65%	0.473	62%	0.386
231	70%	0.494	83%	0.686	53%	0.310	65%	0.435
241	74%	0.557	83%	0.689	70%	0.543	59%	0.386
253	74%	0.533	77%	0.576	49%	0.248	36%	0.044
254	80%	0.649	91%	0.821	71%	0.573	65%	0.471
257	73%	0.529	83%	0.680	64%	0.445	61%	0.380
266	70%	0.459	73%	0.500	73%	0.588	38%	0.048
277	68%	0.397	70%	0.417	50%	0.261	38%	0.075
280	65%	0.389	74%	0.510	56%	0.345	44%	0.193
296	77%	0.601	85%	0.715	59%	0.386	50%	0.224
Avg.	74%	0.530	81%	0.633	61%	0.419	53%	0.282

Table 4.8: Percentage agreement and Cohen's κ between inferred preference judgments from TREC and three kinds of judgments collected via crowdsourcing. For the column of strict preferences, tie judgments in the inferred judgments from TREC are redistributed by assuming different agreement rates. Results under $aar = 50\%$ and 80% are reported.

preferences is 0.419. Both preference judgments agree with TREC significantly better than graded judgments, with p-values from paired sample t-test equal 0.001 and 0.015 respectively. We further compare Cohen's κ from strict preferences ($aar = 50\%$) with the one from weak preferences, getting a p-value from paired sample t-test of 0.004, indicating strict preference judgments agree with judgments from TREC significantly better than weak preferences.

From Table 4.8, it can be seen that agreement from strict preferences under $aar = 50\%$ and weak preferences are 88% and 49% higher than the collected graded judgments in terms of Cohen's κ. We further compare this agreement relative to TREC with the agreement among trained judges reported in literature, similar to (Alonso and Mizzaro, 2012). Intuitively, if agreement between judgments from crowdsourcing and from TREC is comparable to the one among trained judges, we can conclude that judgments from crowdsourcing are good enough to replace those from trained judges. Carterette et al. reported agreement among six trained judges over preference judgments, and the percentage agreement is 74.5% (cf. Table 2 (a) therein), whereas in our work agreement for strict preferences is 74% under $aar = 50\%$ and 81% under $aar = 80\%$. Kazai et al. reported that Fleiss' κ among trained judges over preference judgments is 0.54 (cf. Table 2 PE (e) therein). Thus, we recompute the agreement between strict preference judgments and judgments from TREC in terms of Fleiss' κ, and get $\kappa = 0.504$ under $aar = 50\%$ and 0.637 under $aar = 80\%$. Note that strict preferences are collected in (Carterette et al., 2008) and weak preferences are employed in (Kazai et al., 2013). Since the difference of these two kinds of preference judgments when collected from trained judges is unclear, we regard them the same. We can conclude that the agreement between strict preferences collected via crowdsourcing and TREC are comparable to the one among trained judges. Moreover, compared with strict preference judgments, we can conclude that judgment quality in crowdsourcing is significantly degraded when using weak preferences, which is still significantly better than *graded judgments* as in Table 4.8.

As reported in (Alonso and Mizzaro, 2009) and (Alonso and Mizzaro, 2012), we also observe judges from crowdsourcing can sometimes point out mistakes in TREC judgments. In total, we found around 20 such documents, especially via "test questions", by examining documents (or document pairs) that receive majority judgments opposing the judgments from TREC. One example is clueweb12-0200tw-31-17210[7] and clueweb12-0402wb-10-34906[8] for query 266, "symptoms of heart attack". The former is a news article about a restaurant named "Heart Attack Grill" that provides high calorie food and an incident where a guest had a heart attack; whereas the latter comprehensively includes a list of symptoms that might be signs of heart attack. From TREC, however, these two are labeled as "Relevant" and "Non-Relevant" respectively. Another example is clueweb12-0013wb-31-22050[9] and clueweb12-0806wb-32-26209[10] for query 280, "view my internet history". The former is labeled as "Highly-Relevant" and the latter is labeled as "Relevant" in qrel. However, none of them is relevant: the first page is a comprehensive list about history of internet & W3C, the second page is a question on a forum about how to clean part of internet history.

4.6 Transitivity

In this section, transitivity is examined over both strict and weak preference judgments. Different from (Carterette et al., 2008) and (Rorvig, 1990), we investigate transitivity based on aggregated judgments. This is because the aggregated judgments are the ultimate outcome from crowdsourcing, and also because, as mentioned in Section 4.1, triples from a single judge are too few over individual queries to lead to any conclusions. The results per query are summarized in Table 4.9. It can be seen that over strict preferences, transitivity holds for 96% of triples on average, and the

[7] http://latimesblogs.latimes.com/nationnow/2012/02/diner-appears-to-suffer-heart-attack-at-heart-attack-grill-in-vegas.html

[8] http://my.clevelandclinic.org/services/heart/disorders/coronary-artery-disease/cad-symptoms

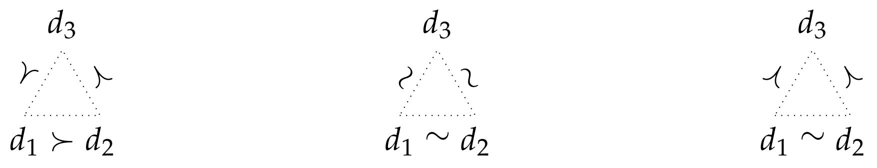
[9] http://vlib.iue.it/history/internet/index.html

[10] http://forum.r-tt.com/cleaning-part-of-internet-history-t178.html

Query	Strict Preferences	Weak Preferences			
	asymTran	*asymTran*	*s2aTran*	*s2sTran*	Overall
216	100% (220/220)	96% (78/81)	89% (90/101)	8% (3/38)	78% (171/220)
222	99% (218/220)	100% (40/40)	98% (117/120)	50% (30/60)	85% (187/220)
226	96% (210/220)	98% (39/40)	87% (86/99)	24% (19/81)	66% (144/220)
231	98% (216/220)	100% (17/17)	95% (107/113)	30% (27/90)	69% (151/220)
241	99% (217/220)	100% (52/52)	99% (112/113)	31% (17/55)	82% (181/220)
253	91% (199/220)	100% (24/24)	86% (66/77)	38% (45/119)	61% (135/220)
254	99% (218/220)	100% (39/39)	97% (105/108)	36% (26/73)	77% (170/220)
257	95% (208/220)	97% (88/91)	86% (87/101)	11% (3/28)	81% (178/220)
266	94% (207/220)	100% (69/69)	98% (123/125)	50% (13/26)	93% (205/220)
277	91% (200/220)	100% (37/37)	82% (109/133)	54% (27/50)	79% (173/220)
280	99% (218/220)	100% (37/37)	85% (85/100)	29% (24/83)	66% (146/220)
296	96% (212/220)	90% (35/39)	77% (82/106)	19% (14/75)	60% (131/220)
Avg.	96% (212/220)	98% (46/47)	90% (98/108)	32% (21/65)	75% (164/220)

Table 4.9: Transitivity over aggregated judgments. The ratio of transitive triples out of triples in different types is reported. The numbers in the bracket are the number of transitive triples divides the total number of triples.

number is between 91% and 100% for individual queries. This number is close to the transitivity reported in (Carterette et al., 2008), where average transitivity is 99% and at least 98% triples from a single judge are transitive. Meanwhile, for weak preferences, this number is only 75% on average, and the minimum percentage is 60% from query 296, indicating that transitivity does not hold in general. To explore the reasons, we further decompose transitivity according to different types of preferences within unique document triples. In particular, the "better than" and "worse than" options are referred to as asymmetric relationships and the "tie" option is referred to as symmetric relationship. As illustrated in Figure 4.7, the transitivity can be categorized as: *asymTran*, which lies among asymmetric relationships (no tie judgment in a triple); *s2aTran*, which lies in between symmetric and asymmetric relationships (only one tie judgment in a triple) and *s2sTran*, which lies among symmetric relationships (at least two tie judgments in a triple). Over each query, the 220 triples are thereby categorized according to the three types on which transitive percentage is computed. From Table 4.9, we can see that *asymTran* holds even better than in strict preferences, meanwhile, *s2aTran* holds for 90% on average. However, over *s2sTran*, the transitivity does not hold anymore: the transitive percentage drops to 32% on average.



(a) *asymTran* (b) *s2sTran* (c) *s2aTran*

Figure 4.7: Different types of transitivity relationships over document triplets.

Time Consumption		average	25th percentile	median	75th percentile
graded judgments	Judgment	2.60	1.37	1.52	1.82
	Total	24.24	11.73	19.55	28.88
strict preferences	Judgment	1.79	1.24	1.37	1.58
	Total	34.17	17.84	25.28	40.98
weak preferences	Judgment	2.07	1.40	1.57	1.91
	Total	32.43	15.77	24.57	39.10

Table 4.10: Average time consumption (in seconds) and quartiles over twelve queries.

We conclude that transitivity holds for over 90% of aggregated strict preference judgments. For weak preference judgments, though, transitivity only holds among non-tie judgments (*asymTran*) and in between tie and non-tie judgments (*s2aTran*). Thus, given judgments $d_1 \sim d_2$ and $d_2 \sim d_3$, we can not infer $d_1 \sim d_3$. We can see that, in terms of transitivity, weak and strict preference judgments behave differently, and extra caution must be taken when assuming transitivity when collecting weak preferences via crowdsourcing.

4.7 Time Consumption

We compare time consumption for different kinds of judgments looking both at total time, which includes the time for reading document(s) and judgment time. The results are summarized in Table 4.10, based on aggregated statistics from twelve queries. For judgment time, it can be seen that judges spend least time with strict preferences. The p-values from two-tailed Student's t-tests between the three kinds of judgments are as follows. P-value equals 0.055 between strict preferences and graded judgments, equals 0.196, between weak preferences and graded judgments, and equals 0.100 between the two kinds of preference judgments. We conclude that judges are slightly but noticeably faster in making judgments with strict preferences than in making the other two kinds of judgments, meanwhile the difference between the time consumption with weak preferences and with graded judgments is insignificant. As for total time, Table 4.10 demonstrates that judges are significantly faster in finishing single graded judgments after considering reading time, with p-value from two-tailed Student's t-test is less than 0.001 relative to both preference judgments. However, there is no significant difference for judges with weak and strict preferences – the corresponding p-value equals 0.168.

Thus, judges are faster in making strict preference judgments. When considering total time, judges need to read two documents in preference judgments, making total time consumption higher. Moreover, when comparing the two kinds of preference judgments, judges take significantly less time with strict preferences, meanwhile there is no difference in terms of total time consumption. Compared with (Carterette et al., 2008) and (Rorvig, 1990), time consumption is measured among judges from crowdsourcing, who are with more diverse reading and judging ability and might be less skillful than trained judges. Actually, the web pages being judged require more than 20 seconds on average to read, making reading time dominate the total time consumption.

4.8 Conclusion

Discussion

It has been demonstrated in Sections 4.5, 4.6 and 4.7 that weak and strict preferences are different in different regards. To investigate the reasons, we further reduce the number of options in weak preferences by merging "tie" with "A is better", merging "tie" with "B is better" or merging the two non-tie options, measuring the agreement among judges, getting Fleiss' $\kappa = 0.247, 0.266$, and 0.073 respectively. Recall that the Fleiss's κ equals 0.253 in Table 4.4 when three options are considered. The corresponding p-values from two-tailed Student's t-tests relative to the one with three options are 0.913, 0.718, and less than 0.001. It can be seen that judges tend to disagree more when making choices between ties and non-tie judgments. Put differently, the threshold to make a non-tie judgment is ambiguous and is varied among different judges. This implies that the tie option actually makes the judgments more complicated, namely, judges have to firstly determine whether the difference is large enough to be non-tied before judging the preferences.

Despite the relatively low transitivity observed for weak preference judgments in Section 4.6, it can be seen from Table 4.9 that, the average percentage for *asymTran* and *s2aTran* are larger or equal than 90%. And it is the *s2sTran* on which the judgments become intransitive. In other words, the transitivity still holds when assessors judge preferences between documents that have significant difference in relevance, whereas intransitivity happens when the relevance of two documents gets close. This actually corresponds to our arguments from Section 4.1, that it is unlikely that each of the possibly hundreds of returned documents has its own degree of relevance, indicating that assessors tend to disagree with each other about tie judgments when the difference is tiny. On the other hand, in both *Quick-Sort-Judge* (Song et al., 2011) and the *Merge-Tie-Judge* introduced in 4.3.2, only the *asymTran* and *s2aTran* help in determining the preference ranking among documents, whereas *s2sTran* contributes to the growth of tie partitions. Moreover, Table 4.7 demonstrates that, when only considering non-tie judgments, the agreement of weak preference judgments relative to the ground-truth is still higher than the one from graded judgments in Table 4.5.

Therefore, considering the potential of ties in reducing the number of judgments, as demonstrated in Section 4.3, we argue that the ties should be used to reduce the number of judgments by employing the proposed *Merge-Tie-Judge* mechanism, meanwhile sacrificing certain quality.

Conclusion

Finally, we answer the four research questions from Section 4.1 to summarize this chapter.

> **RQ1**: *Whether ties can be used to reduce the number of judgments?*
> **Answers**: We demonstrate that ties can dramatically reduce the number of judgments. And we propose a novel judgment mechanism, coined *Merge-Tie-Judge*, which is more robust, and can further reduce the number of judgments significantly by utilizing the "cluster effect".
>
> **RQ2**: *Can weak/strict preference judgments collected using crowdsourcing replace judgments by trained judges in* TREC*?*
> **Answers**: We conclude that transitivity holds for over 90% aggregated strict preference judgments. For weak preference judgments, the transitivity holds among non-tie judgments (*asymTran*) and in between tie and non-tie judgments (*s2aTran*).
>
> **RQ3**: *Do weak/strict preference judgments exhibit transitivity when collected using crowdsourcing?*
> **Answers**: Judges are faster in making strict preference judgments. When comparing the two kinds of preference judgments, judges take significantly less time with strict preferences, meanwhile there is no difference in terms of total time consumption.
>
> **RQ4**: *How do weak/strict preference judgments compare against graded judgments in terms of time consumption?*
> **Answers**: Both strict and weak preferences collected via crowdsourcing are better than the collected graded judgments. And only the one from strict preference judgments is comparable to the manual judgments from TREC.

5 RE-PACRR: A Novel Neural IR Model

5.1 Introduction

Despite the widespread use of deep neural models across a range of linguistic tasks, whether such models can improve information retrieval (IR) and what components an IR retrieval model should include remain open questions. In standard ad-hoc IR, the goal is to produce a ranking of relevant documents given an open-domain (“ad hoc”) query and a document collection. Many early neural IR models can be categorized as *semantic matching* models, as they embed both queries and documents into a low-dimensional space, and then compute matching signals based on such dense representations, often via cosine similarity. Examples in this regard include *DSSM* (Huang et al., 2013) and *C-DSSM* (Shen et al., 2014). The notion of relevance is inherently asymmetric, however, making it different from well-studied semantic matching tasks such as semantic relatedness and paraphrase detection. Instead, *relevance matching* models such as MatchPyramid (Pang et al., 2016a) and DRMM (Guo et al., 2016) resemble traditional IR retrieval measures in that they directly consider the relevance of the document contents with respect to the query. The two classes of models are fairly distinct, and in this chapter we focus on relevance matching models. The more recent DUET model (Mitra et al., 2017) is a hybrid approach that combines signals from a local model for relevance matching and a distributed model for semantic matching.

Given that relevance matching approaches mirror ideas from traditional retrieval models, the decades of research on ad-hoc IR can guide us with regard to the specific kinds of relevance signals a model ought to capture. Unigram matches are the most obvious signals to be modeled, as a counterpart to the term frequencies that appear in almost all traditional retrieval models. Beyond this, positional information, including where query terms occur and how they depend on each other, can also be exploited, as demonstrated in retrieval models that are aware of term proximity (Tao and Zhai, 2007) and term dependencies (Huston and Croft, 2014; Metzler and Croft, 2005). Beyond the positional information, there still exist multiple factors that have been demonstrated to be relevant to the performance of a retrieval model, including the disambiguation of the term occurrences and the density of the relevance information etc.. Thus, to the end of this chapter, we also investigate several components to specially cater for factors beyond the positional information.

Unigram signals are directly taken as input by DRMM (Guo et al., 2016) and subsequently summarized as histograms, where the experiments suggest that DRMM compares favorably to previously proposed Neural retrieval models, and that unigram matching signals are well-exploited. As for positional information, both the MatchPyramid (Pang et al., 2016a) and local DUET (Mitra et al., 2017) models account for it by incorporating convolutional layers based on similarity matrices between queries and documents. Although this leads to more complex models, MatchPyramid performs significantly worse than DRMM (Guo et al., 2016), while DUET's local model performs similarly to DRMM (Mitra et al., 2017). This indicates the difficulty of going beyond unigrams to utilize positional information in deep neural IR models. Intuitively, unlike in standard sequence-based models, the interactions between a query and a document are sequential along the query axis as well as along the document axis, making the problem multi-dimensional in nature. In addition, this makes it non-trivial to combine matching signals from different parts of the documents and over different query terms. In fact, we argue that both MatchPyramid and local DUET fail to fully account for all factors. To address these shortcomings, we conjecture that a suitable combination of convolutional kernels and feedforward layers can lead to a model that better accounts for these factors. In particular, we present a novel re-ranking model called *PACRR* (*Position-Aware Convolutional Relevance Retrieval* Model). Our approach first produces similarity matrices that record the semantic similarity between each query term and each individual term occurring in a document. These matrices are then fed through a series of convolutional, max-k-pooling, and feedforward layers so as to capture interactions corresponding to, for instance, bigram and trigram matches, and finally to aggregate the signals in order to produce global relevance assessments. In our model, the convolutional layers are designed to capture both unigram matching and position information over text windows with different lengths; k-max pooling layers are along the query dimension, preserving matching signals over different query terms; several feedforward layers combine signals from different query terms to produce a query-document relevance score.

As the second part in this chapter, we further highlight several important aspects based on past research within the IR community, and propose several deep neural components for PACRR, hoping to address these factors better. These components are integrated to form a novel neural model called RE-PACRR, short for Relevance Enhanced PACRR, which builds on PACRR. However, the insights we identify and components we propose are not specific to PACRR, and we expect them to be similarly relevant to other state-of-the-art neural models, such as DUET (Mitra et al., 2017), DRMM (Guo et al., 2016), and MatchPyramid (Pang et al., 2016a).

Vocabulary mismatch suggests that one not only relies on query terms, but also consider semantically similar terms in the relevance matching – as best exemplified by query-expansion methods (Cronen-Townsend et al., 2004; Xu and Croft, 1996). For example, given query "*global* *automobile market share 2016*", the query term could be expanded with the synonyms of "automobile", such as, "car" and "auto". Such issues are naturally addressed by the similarity among word vectors of different terms in recent neural IR models, as in (Guo et al., 2016; Hui et al., 2017b; Mitra et al., 2017; Pang et al., 2016a). However, whether the established word embedding is good enough to encode the desired similarity is questionable (Zamani and Croft, 2017), and we attempt to verify whether such similarity is always helpful compared with an exact match as in the local model in DUET model (Mitra et al., 2017).

Disambiguation refers to distinguishing different senses of the same term. For retrieval models such as BM25 that rely on exact term matching, some of the matches may actually stem from different senses of the same term in a query and a document. Such sense mismatches are incorrectly considered as relevance matches. For example, given the query "*Jaguar SUV price*", the term "Jaguar" refers to a car brand, but the term "jaguar" could also refer to an animal, as in the following sentence.

In a safari wild park near London, an attack happened when a jaguar was irritated by a horn from a tourist SUV, which, fortunately, caused no harm.

Therefore, when evaluating the relevance between this sentence-query pair, the exact term match of "jaguar" should not be counted as a relevance match. Models such as DUET (in its distributed component) may be able to account for this by encoding a query and a document into a dense vector space. Most other neural IR models do not explicitly account for disambiguation. In this work, a context checking module is proposed and plugged into the PACRR model, allowing the model to consider sense matching as well as term matching.

Proximity takes into account information about where query terms occur in a document and how they depend on each other – as exploited by retrieval models aware of term proximity (Tao and Zhai, 2007). Term proximity emphasizes the frequent co-occurrence of different query terms within a small text window, which are not necessarily successive. Intuitively, a small text window including multiple different query terms is more likely to be relevant in the sense that it covers more relevant information to the query. For example, given the query "*Jaguar SUV price*", a text window including all three terms is more likely to be relevant, since both "Jaguar SUV" and "price" are included. To consider proximity, a model needs to be capable of capturing local matching information beyond a single query term. In this regard, it has been partially considered in MatchPyramid (Pang et al., 2016a) and PACRR, but these models do not consider proximity of all query terms at once. In this work, we further investigate a new building block to model proximity by including a sufficiently large kernel to cover all query terms, which makes it possible to fully model proximity.

Query coverage aims at comprehensively catering to all relevant concepts in the query, avoiding to bias the results towards only individual concepts occurring in the query. Take, for instance, a verbose query such as the following (Gupta and Bendersky, 2015):

Provide information on all kinds of material international support provided to either side in the Spanish Civil War.

Both "Spanish" and "civil war" should likely be mentioned in this case. More generally, a relevant document to a query should, to some degree, cover all of the most salient concepts in it. Existing neural IR models satisfy query coverage when combining the matching signals from different query terms. We argue that such a combination should be position-independent. Put differently, the way to combine the relevance signals and the weights allocated to different query terms should only depend on the inherent properties of the query terms and their relevance signals. Such matching signals already cover both unigrams and ngrams (e.g., in PACRR the matching signal for the term "civil" includes both the unigram signal and bigram matching signal for "civil war"), so the relative positions of these matching signals need not be considered when the signals are combined. This should particularly be true when the inputs are zero-padded in the tail of a query, where a model is not supposed to learn combination weights based on the occurrences of zeros in the tail positions. In this regard, we argue that the positional independence is not fully implemented in established model architectures. To address this, we introduce a random permutation phase before combining signals, removing the position-dependent factors and thereby improving the generalization ability of the model.

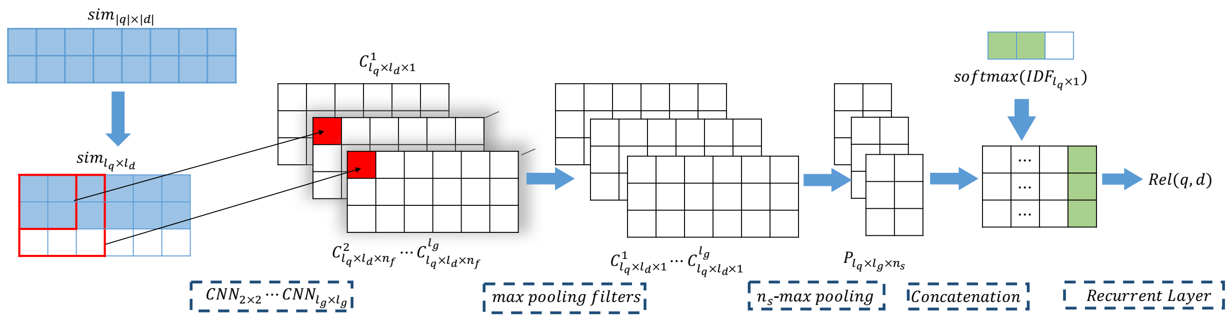
Cascade reading model. In addition to the amount of relevant information in a document, a model should also consider the density of the relevant information. An extreme example is the comparison between one relevant document and a pseudo-document composed of several concatenated non-relevant documents with the same relevant document appended at the end. Although the same amount of relevance matches are included in the original relevant document and the pseudo-document, one would expect that the original document is preferred, due to the relevant information being presented earlier and overall more densely. Beyond this, according to the cascade model from (Craswell et al., 2008), the gains of the relevance in a document should be degraded when enough relevant information has been observed earlier, since a user may already stop reading after collecting enough relevant information from the current document. Such signals might be learned implicitly by a $1 \times doclen$ kernel in the local matching of DUET. However, to the best of our knowledge, it has not been explicitly modeled in established neural IR models. To address this, we propose a cascade component to model the position of the relevant signals in addition to the total amount of relevant signals.

Contributions. In this chapter, we firstly propose a novel position-aware neural IR model, named PACRR, to encode positional information. Beyond that, a novel model, named RE-PACRR, is proposed by incorporating all of the aforementioned better-designed building blocks: the vocabulary mismatching module, the context checking module, the term proximity kernel, the signal combination regularizer, and the cascade component. The proposed PACRR and RE-PACRR models are examined through extensive experiments based on relevance judgments from TREC, and compared with multiple state-of-the-art models including MatchPyramid, DRMM, the local model in DUET, and the PACRR model. We find that both PACRR and RE-PACRR compare favorably against the baseline models. Remarkably, when re-ranking the search results from a naïve ranker with RE-PACRR, namely a query-likelihood ranking model, the re-ranked runs are ranked top-1 under both ERR@20 and nDCG@20 in the Web Track 2012–14, improving both evaluation measures by 100% on average. Finally, in favor of a better understanding of different building blocks, ablation studies are conducted in the end, by adding one component in at a time to the PACRR model to investigate the effects of the corresponding components.

5.2 Related Work

Ad-hoc retrieval systems aim at ranking documents with respect to their relevance relative to given queries. Recently, the promise of deep learning as a potential driver for new advances in retrieval quality has attracted significant attention. Early works like Deep Structured Semantic Models (Huang et al., 2013) (*DSSM*) learned low-dimensional representations of queries and documents in a shared space and performed ranking by comparing the cosine similarity between a given query's representation and the representations of documents in the collection. Similar approaches such as C-DSSM (Shen et al., 2014) relied on alternative means of learning document representations. Severyn and Moschitti further combined learned semantic representations with external features to rank question answers and tweets. In addition, other approaches to learning representation of documents or queries can also be deployed for retrieval, by taking the similarity between such representations as relevance scores. Models of this sort include ARC-I and ARC-II by (Pang et al., 2016b), which performed text classification by producing representations of documents and queries separately (ARC-I), or, alternatively, produced representations that also considered the interactions between queries and documents jointly (ARC-II). The experiments from (Guo et al., 2016) (as well as our own pilot experiments) showed that none of the above deep models can consistently improve traditional retrieval models such as *QL* on the long-established TREC benchmarks.

Guo et al. highlighted that the matching needed for information retrieval differs from the kind that is used in NLP tasks, which typically aim at *semantic matching*. Information retrieval ranking models, in contrast, are concerned with *relevance matching*. The former focuses on comparing the meaning of two input texts, while the latter focuses on the inherently asymetric goal of representing the text and determining its relevance to a user query. The overall semantics of the document needn't be similar to the user intent expressed by the query. Indeed, traditional retrieval models such as query-likelihood (*QL*) (Zhai and Lafferty, 2001) are heavily based on this notion of relevance matching, capturing it via unigram occurrences of query terms. *DRMM* learns the same sort of unigram matches as in traditional retrieval models, but with more advanced instruments from deep learning. In particular, *DRMM* takes a sequence of fixed-length query term similarity histograms as input; each histogram h_j represents the matches between one query



$sim_{|q|\times|d|}$
$sim_{l_q\times l_d}$
$C^1_{l_q\times l_d\times 1}$
$C^2_{l_q\times l_d\times n_f}\cdots C^{l_g}_{l_q\times l_d\times n_f}$
$C^1_{l_q\times l_d\times 1}\cdots C^{l_g}_{l_q\times l_d\times 1}$
$P_{l_q\times l_g\times n_s}$
$softmax(IDF_{l_q\times 1})$
$Rel(q,d)$
$CNN_{2\times 2}\cdots CNN_{l_g\times l_g}$
max pooling filters
n_s-max pooling
Concatenation
Recurrent Layer

Figure 5.1: The pipeline of *PACRR*.

term q_j in a given query q and the terms in a given document. The query similarity histograms are each fed through a series of fully connected layers to produce a similarity signal for each query term. The document's relevance score $rel(q,d)$ is a weighted summation of each query term's similarity signal. *DRMM* was evaluated on TREC Robust Track 2004 and the Web Track 2009–11. The experiments demonstrated that *DRMM* outperformed both *DSSM* and several deep models developed for NLP tasks. In fact, *DRMM* outperformed the strongest *DSSM* variant by a minimum of 68% nDCG@20, and we observed similar results in our pilot experiments. Another recent work called DUET (Mitra et al., 2017) proposes a deep ranking model that considers both exact matches between document and query terms (the *local model*) and the similarity between low-dimensional representations of the query and document (the *distributed model*). The authors evaluated DUET on queries from Bing search logs with 199,753 training examples, and reported nDCG@10 increases over *DRMM* of 18% (weighted by query occurrences) and 2% (unweighted). Interestingly, *DRMM* did not substantially outperform *QL* in this comparison, whereas Guo et al. saw large increases in performance in their experiments on TREC Web Track data. We remark that the weighted evaluation set appears incomparable with the TREC Web Track. In particular, the test queries from the TREC Web Track include a significant proportion of more focused topics designed to represent more specific, less frequent, and possibly more difficult queries (Collins-Thompson et al., 2015b), whereas the weighted test data places more emphasis on frequent head queries. Given that the 1 million documents and 200,000 Bing queries used to train DUET are far beyond what is available to us, we compare our methods with the local model of DUET.

5.3 (RE-)PACRR: A Novel Neural IR Model

In this section, we firstly describe our novel neural IR framework, named PACRR. Thereafter, several better-designed components are plugged-in to boost the proposed PACRR in Section 5.3.2, ending up with RE-PACRR.

5.3.1 PACRR

In this section, we introduce the PACRR model, which is summarized in Figure 5.1. In general, PACRR takes a similarity matrix between a query q and a document d as input, and the output of the model is a scalar, namely, $rel(d, q)$, indicating the relevance of document d to query q. PACRR attempts to model query-document interactions based on these similarity matrices.

Relevance matching. We propose encoding the query-document relevance matching via query-document similarity matrices $sim_{|q|\times|d|}$ that encode the similarity between query and document terms. In particular, given a document d and a query q, the similarity between every term pair from d and q is encoded as a similarity matrix $sim_{|q|\times|d|}$, where sim_{ij} corresponds to the similarity between the i-th term from the query q and the j-th term from the document d. When using cosine similarity, we have $sim \in [-1, 1]^{|q|\times|d|}$. Our similarity matrix approach retains a richer signal than the similarity histogram approach used in prior work (Guo et al., 2016), which is limited to performing relevance matching against unigrams. Our matrices preserve both n-gram relevance signals and query coverage information. In particular, n-gram matching corresponds to consecutive document terms that are highly similar to at least one of the query terms, while query coverage is reflected in the number of rows in *sim* that include at least one cell with high similarity. As in (Guo et al., 2016), the similarity between a query term q and document term d is calculated by taking the cosine similarity between the terms' *word2vec* vectors.

The subsequent processing in PACRR's convolutional layers requires that each query-document similarity matrix have the same dimensionality. Given that the lengths of queries and documents vary, we first transform the raw similarity matrices $sim_{|q|\times|d|}$ into $sim_{l_q\times l_d}$ matrices with uniform l_q and l_d as the number of rows and columns. We unify the query dimension l_q by zero padding it to the maximum query length. With regard to the document dimension l_d, we employ *firstk* strategy. Namely, zero pad the document dimension of each similarity matrix to the maximum document length. Akin to (Mitra et al., 2017), the *firstk* distillation method simply keeps the first k columns in the matrix, which correspond to the first k terms in the document. If $k > |d|$ the remaining columns are zero padded. The *firstk* method is equivalent to zero padding if k is set to the maximum document length.

Convolutional n-gram relevance matching. The purpose of this step is to match query n-grams with document n-grams given a query-document similarity matrix as input. This is accomplished by applying multiple two-dimensional convolutional layers with different kernel sizes to the input similarity matrix. Each convolutional layer is responsible for a specific n-gram size; by applying its kernel on $n \times n$ windows, it produces a similarity signal for each window. When the *firstk* method is used, each convolutional layer receives the same similarity matrix $sim_{l_q\times l_d}$ as input because *firstk* produces the same similarity matrix regardless of the n-gram size. We use $l_g - 1$ different convolutional layers with kernel sizes $2 \times 2, 3 \times 3, \ldots, l_g \times l_g$, corresponding to bi-gram, tri-gram, ..., l_g-gram matching, respectively, where the maximum n-gram size to consider is governed by a hyper-parameter l_g. The original similarity matrix corresponds to unigram matching, while a convolutional layer with kernel size $n \times n$ is responsible for capturing n-gram matching for an $n \times n$ square within $sim^n_{l_q \times l_d}$. Each convolutional layer applies n_f different filters to its input, where n_f is another hyper-parameter. We use a stride of size (1, 1) for the *firstk* distillation method, meaning that the convolutional kernel advances one step at a time in both the query and document dimensions. Thus, we end up with $l_g - 1$ matrices $\mathcal{C}^n_{l_q \times l_d \times n_f}$, where n indicates the corresponding n-gram size. The original similarity matrix will be used to handle unigrams.

Max pooling. The purpose of this step is to capture the n_s strongest similarity signals for each query term. Measuring the similarity signal separately for each query term allows the model to consider query term coverage, while capturing the n_s strongest similarity signals for each query term allows the model to consider signals from different kinds of relevance matching pattern, e.g., n-gram matching and non-contiguous matching. In practice, we use a small n_s to prevent the model from being biased by document length; while each similarity matrix contains the same number of document term scores, longer documents have more opportunities to contain terms that are similar to query terms. To capture the strongest n_s similarity signals for each query term, we first perform max pooling over the filter dimension n_f to keep only the strongest signal from the n_f different filters. We then perform k-max pooling (Kalchbrenner et al., 2014) over the query dimension l_q (corresponding to rows in the original similarity matrix) to keep the strongest n_s similarity signals for each query term. Both pooling steps are performed on each of the $l_g - 1$ matrices $\mathcal{C}^i$ from the convolutional layer (corresponding to each n-gram size) and on the original similarity matrix (corresponding to unigrams) to produce the 3-dimensional tensor $\mathcal{P}_{l_q \times l_g \times n_s}$. This tensor contains the n_s strongest similarity signals for each query term and for each n-gram size across all n_f filters.

Combinations of signals from different query terms. According to our pilot experiments, the performance of the model does not change significantly when using a LSTM layer as in (Hui et al., 2017b) or with a stack of dense layers, which have been demonstrated to be able to simulate an arbitrary function (Goodfellow et al., 2016). Such dense layers can easily learn in parallel, leading to faster training, whereas back-propagation through an LSTM layer is much more expensive due to its sequential structures. From Section 5.4.2, it can be seen that efficiency is very important for this study, due to the huge amount of model variants to be trained and the limited availability of hardware at our disposal. Thus, the signals in $P_{l_q \times (l_g n_s)}$, together with the inverse document frequency for individual query terms, are fed into a feedforward network with two dense layers, generating the final query-document relevance score $rel(d, q)$.

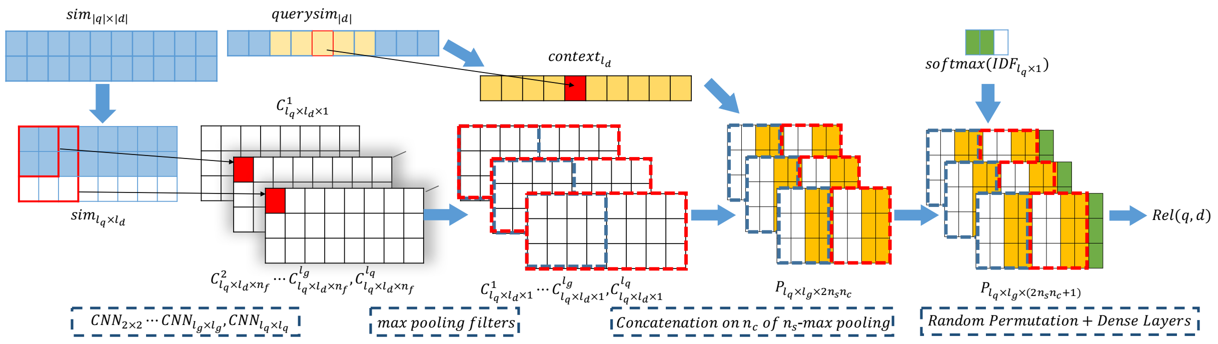

$sim_{|q|\times|d|}$
$querysim_{|d|}$
$context_{l_d}$
$softmax(IDF_{l_q\times 1})$
$C^1_{l_q\times l_d\times 1}$
$sim_{l_q\times l_d}$
$Rel(q,d)$
$C^2_{l_q\times l_d\times n_f} \cdots C^{l_g}_{l_q\times l_d\times n_f}, C^{l_q}_{l_q\times l_d\times n_f}$
$C^1_{l_q\times l_d\times 1} \cdots C^{l_g}_{l_q\times l_d\times 1}, C^{l_q}_{l_q\times l_d\times 1}$
$P_{l_q\times l_g\times 2n_s n_c}$
$P_{l_q\times l_g\times (2n_s n_c+1)}$
$CNN_{2\times 2} \cdots CNN_{l_g\times l_g}, CNN_{l_q\times l_q}$
max pooling filters
Concatenation on n_c of n_s-max pooling
Random Permutation + Dense Layers

Figure 5.2: The pipeline of *RE-PACRR*.

5.3.2 RE-PACRR

In this section, we further describe the novel components in the RE-PACRR model. The architecture of this model is summarized in Figure 5.2.

Binary vs real value similarity matrix: vocabulary expansion. As a building block mainly serving as examination purpose, we simply employ either binary value or the real value in the input similarity matrix.

Context checking module: disambiguation of the relevance matching. Beyond the $sim_{l_q\times l_d}$ from PACRR, we introduce another input matrix denoted as $querysim_{|d|}$, recording the similarity for each term relative to the query. In particular, the vector of a query is computed by averaging the word vectors of all query terms, and cosine similarity is computed between each term vector in the document relative to this query vector. As in (Hui et al., 2017b), we use pre-trained word2vec[1] embeddings to compute term similarities due to its availability. In the future, one may desire to replace this with other embeddings such as the dual embedding from (Mitra et al., 2016) and the relevance-based embedding from (Zamani and Croft, 2017). Thereafter, given a term at position i, namely, $querysim[i]$, the similarity for its context is computed by averaging the similarity in its surrounding windows, resulting in $context_{l_d}$ after truncating or zero-padding in accordance with l_d. Thus, for a term at position i in a document, its context similarity $context[i]$ is computed as

$$context[i] = \frac{\sum_{j\in[i-w,i+w]} querysim[i]}{2*w+1} \; .$$

In the model, to avoid relevance matching due to ambiguity as in the "jaguar" example from Section 5.1, intuitively, only when both the term and its context are highly similar with a query term (or n-gram in the query) do matching signals represent a correct relevance match. Thus, when combining the top-n_s signals from individual query terms, the corresponding similarities

[1]https://code.google.com/archive/p/word2vec/

for these top-n_s signals from $context_{l_d}$ are also concatenated, making the matrices $P_{l_q \times (l_g n_s)}$ become $P_{l_q \times (2 l_g n_s)}$. This enables the aggregating model, namely, a feed-forward network, to take the ambiguity into account when determining the ultimate score. For example, in the "jaguar" example in Section 5.1, in the context of "jaguar" there are "safari" and "wild park", which have low similarity with query terms such as "SUV" and "price", effectively making the model understand that the two "jaguar" occurrences in the query and in the sentence are referring to different senses.

Model proximity with a larger CNN kernel. As discussed in Section 5.1, we assume that proximity could be modeled over the entire query, instead of over multiple small text pieces in the query, as in MatchPyramid and PACRR. Thus we propose to implement an extra kernel with size $l_q \times l_q$, adding it along with other CNN kernels. This leads to an extra matrix after the CNN layer, namely $C^{l_q}_{l_q \times l_d \times l_f}$, and ultimately an extra matrix after the max-pooling of filters, namely, $C^{l_q}_{l_q \times l_d \times 1}$. Thereafter, such proximity signals are passed on to the follow-up pipeline in the model, resulting in a matrix $P_{l_q \times ((l_g+1) n_s)}$ before the combination.

Cascade k-max: simulating the cascade reading approach. As discussed in Section 5.1, not only the strength but also the offsets of relevance signals matter. We propose to encode such cascade factors by conducting k-max pooling at multiple offsets in a document, instead of pooling only on the entire document. For example, one could conduct multiple k-max pooling at 25%, 50%, 75%, and 100% of a document, ending up with $P_{l_q \times (4 l_g n_s)}$. This corresponds to when a user sifts through a document and evaluates its relevance after finishing the first, second, third, or fourth quarter of the document. The list of offsets at which cascade k-max pooling is conducted is governed by an array $cpos$, e.g., $cpos = [25\%, 50\%, 75\%, 100\%]$ in the above example, and the length of this array is denoted as n_c, which are both hyper-parameters.

Randomly permute query terms before feed-forward layer: improving generalization. As mentioned in Section 5.1, the combination of relevance signals among different queries is position-independent. In light of this, we propose to randomly permute rows in $P_{l_q \times (l_g n_s)}$ before aggregating them. Note that each row contains signals for multiple n-gram lengths; permuting the rows does not prevent the model from recognizing n-grams. We argue that, taking advantage of this independence, the permutation can effectively improve the generalization ability of the model, making the computation of the relevance scores depend solely on the importance of a query term (idf) and the relevance signals aggregated on it. This should be particularly helpful when training on short queries ($|q| < l_q$), where padded zeros are in the tail of $sim_{l_q \times l_d}$. Without permutation, a model might remember that the relevance signals at the tail of a query, e.g., the several final rows in $sim_{l_q \times l_d}$, contribute very little and are mostly zero, leading to it mistakenly degrading the contribution from terms at tail positions when inferring relevance scores for longer queries.

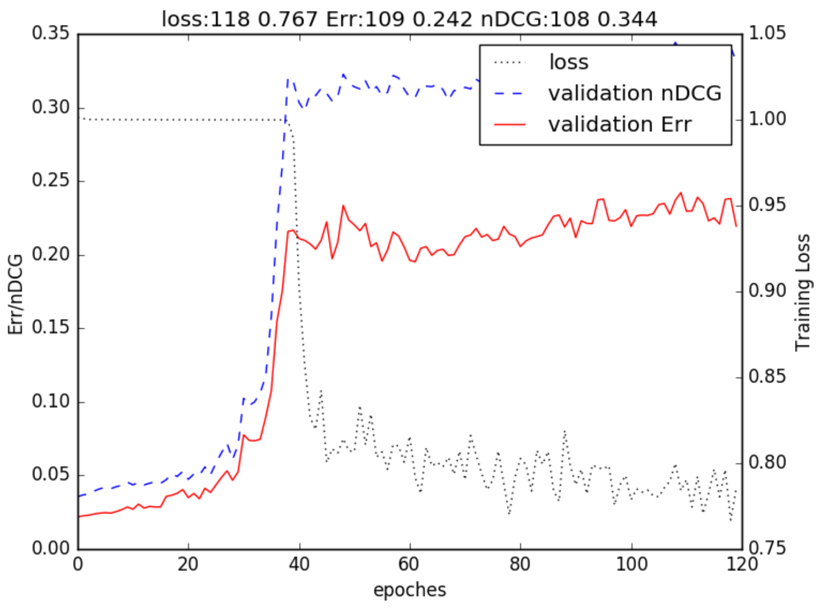


Figure 5.3: The loss on training data and the Expected reciprocal rank (ERR@20) and nDCG@20 per epoch on validation data when training on Web Track 2010–14. The x-axis denotes the epoch. The y-axis indicates the ERR@20/nDCG@20 on the left and the training loss on the right. The best performance appears on 109th epoch with ERR@20=0.242. The lowest loss (0.767) on the training data occurs after 118 epochs.

5.3.3 Training objective and prediction

Our model is trained on triples consisting of a query q, relevant document d^+, and non-relevant document d^- using stochastic gradient descent (SGD) to minimize a cross entropy loss as in Equation 5.1. According to (Dehghani et al., 2017), it has been demonstrated that a cross-entropy loss may lead to better performance than the hinge loss.

$$\mathcal{L}(q,d^+,d^-;\Theta) = -log\frac{\exp(rel(q,d^+))}{\exp(rel(q,d^+)) + \exp(rel(q,d^-))} \tag{5.1}$$

At each training step, we perform SGD on a mini-batch of 16 triples. For the purpose of choosing the triples, we consider all documents that are judged with a label more relevant than *Rel*[2] as *highly relevant*, and put the remaining relevant documents into a *relevant* group. To pick each triple, we sample a relevance group with probability proportional to the number of documents in the group within the training set, and then we randomly sample a document with the chosen label to serve as the positive document d^+. If the chosen group is the highly relevant group, we randomly sample a document from the relevant group to serve as the negative document d^-. If the chosen group is the relevant group, we randomly sample a non-relevant document as d^-. This

[2]Judgments from TREC include junk pages (*Junk*), non-relevance (*NRel*), relevance (*Rel*), high relevance (*HRel*), key pages (*Key*) and navigational pages (*Nav*).

sampling procedure ensures that we differentiate between highly relevant documents (i.e., those with a relevance label of *HRel*, *Key* or *Nav*) and relevant documents (i.e., those are labeled as *Rel*). The training continues until a given number of epochs is reached. The model is saved at every epoch. We use the model with the best ERR@20 on the validation set to make predictions. One example for this training procedure is displayed in Figure 5.3.

5.4 Experiment

In this section, we empirically compare the proposed PACRR and RE-PACRR with multiple state-of-the-art neural IR models using manual relevance judgments from six years of the TREC Web Track. The comparison is based on three benchmarks, namely, re-ranking search results from a simple initial ranker, coined RERANKSIMPLE, re-ranking all runs from the TREC Web Track, coined RERANKALL, and further examining the classification accuracy of the neural IR models in determining the order of document pairs, coined PAIRACCURACY. We compare our model with multiple state-of-the-art neural IR models including DRMM (Guo et al., 2016), DUET (Mitra et al., 2017) and MatchPyramid (Pang et al., 2016a). As our focus is on the deep relevance matching model, we only compare against DUET's local model, denoted as DUETL. Additionally, the TREC benchmarks contain much less data than that used to train DUET's distributed model in (Mitra et al., 2017).

5.4.1 Experimental Setup

We rely on the 2009–2014 TREC Web Track ad-hoc task benchmarks[3], which are based on the CLUEWEB09 and CLUEWEB12 datasets as document collections. In total, there are 300 queries and more than 100k judgments (qrels). Three years (2012–14) of query-likelihood baselines (*QL*) (Terrier (Ounis et al., 2006) version without filtering spam) provided by TREC[4], and the search results from runs submitted by participants from each year are both employed as initial rankings in the RERANKSIMPLE and RERANKALL benchmarks, respectively. In total, there are 71 (2009), 55 (2010), 62 (2011), 48 (2012), 50 (2013), and 27 (2014) runs. ERR@20 (Chapelle et al., 2009) and nDCG@20 (Järvelin and Kekäläinen, 2002) are employed as evaluation measures, and both are computed with the script from TREC[5].

Training. Models are trained and tested in a round-robin manner, using individual years as training, validation and test data. Specifically, the available judgments are considered in accordance with the individual years of the Web Track, with 50 queries per year. Proceeding in a round-robin manner,

[3]`http://trec.nist.gov/tracks.html`
[4]`https://github.com/trec-web/trec-web-2014`
[5]`http://trec.nist.gov/data/web/12/gdeval.pl`

we report test results on one year by using combinations of every four years and the left-out year in the remaining years for training and validation, respectively. Model parameters and the number of training epochs are chosen by maximizing the ERR@20 on the validation set for each training validation combination separately. Thereafter, the selected model is used to make predictions on the test data. Hence, for each test year, there are five different predictions each from a training and validation combination. Akin to the procedure in cross-validation, we report the average of these five test results as the ultimate result for individual test years, and conduct a Student's t-test over them to determine whether there is a statistically significant difference between different methods. For example, a significant difference between two evaluated methods on a particular test year is claimed if there exists a significant difference between the two vectors with five scores for individual methods. We argue that, this way, the effects of the choice of training and validation data are minimized. This was motivated by an observation that the closeness of the subsets for training and for validation can adversely influence the model selection. For example, when using a validation set containing queries from the same TREC years as in the training set, a model could sometimes over-fit both the training and validation sets at the same time, leading to poor results on the test set.

Choice of the hyper parameters. After selecting hyper parameters on the validation dataset, for the PACRR model, the dimensions are selected as $l_d = 800$ and $l_q = 16$, the k-max pooling size is $n_s = 3$, the maximum n-gram size is $l_g = 3$, and the number of filters used in convolutional layers is $n_f = 32$. For the new components in RE-PACRR, we fix the size of the context window as 4 on both sides, leading to a context vector computed by averaging 9, namely, $4 + 4 + 1$, surrounding similarity values in *querysim*. For the cascade component, we conduct k-max pooling with $cpos = [25\%, 50\%, 75\%, 100\%]$. DRMM (DRMM$_{LCH \times IDF}$), DUETL, and MatchPyramid (Pang et al., 2016a) are trained similarly. The size of the dense layer for DRMM is fixed to 5 following the setting in (Guo et al., 2016); the document dimension for both DUETL and MatchPyramid is also set to 800. All these methods are trained with a cross-entropy loss as summarized in Equation 5.1. Beyond that, we employ the batch size to 16 and we train individual models to at most 150 epochs for all comparing models.

5.4.2 Results for (RE-)PACRR

RERANKSIMPLE. We first examine how well the proposed model performs when re-ranking the search results from a basic initial ranker, the query-likelihood model. The ultimate quality of the re-ranked search results depends on both the strength of the initial ranker and the quality of the re-ranker. The query-likelihood model, as one of the most widely used retrieval models, is used due to its efficiency and practical availability, given that it is included in most retrieval toolkits as a default model. The results are summarized in Tables 5.1 and 5.2. The absolute scores in terms of ERR@20 and nDCG@20 are reported, together with the improvements relative to the measures on

Measures	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
ERR@20	wt12	0.390 (p↑D↑L↑M↑) 121% 1	0.347 (d↑L↑M↑) 96% 1	0.309 (P↓L↑) 74% 5	0.218 (P↓D↓M↓) 23% 18	0.314 (p↓L↑) 78% 4
	wt13	0.190 (p↑D↑L↑M↑) 89% 1	0.175 (D↑L↑M↑) 74% 3	0.135 (P↓D↓) 34% 15	0.137 (P↓D↓) 36% 14	0.155 (P↓L↑M↑) 54% 7
	wt14	0.246 (P↑D↑L↑M↑) 88% 1	0.223 (D↑L↑M↑) 70% 1	0.183 (P↓) 40% 12	0.174 (P↓D↓) 33% 16	0.195 (P↓L↑) 49% 8

Table 5.1: ERR@20 on TREC Web Track 2012–14 when re-ranking search results from *QL*. The comparisons are conducted between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach. In addition, the relative improvements (%) and ranks among all runs within the respective years according to ERR@20 are also reported after the absolute scores.

Measures	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
nDCG@20	wt12	0.281 (P↑D↑L↑M↑) 163% 1	0.248 (D↑L↑M↑) 133% 2	0.212 (P↓L↑) 99% 5	0.155 (P↓D↓M↓) 45% 18	0.205 (P↓L↑) 93% 6
	wt13	0.345 (p↑D↑L↑M↑) 82% 1	0.324 (D↑L↑M↑) 70% 2	0.263 (P↓L↑) 39% 6	0.239 (P↓D↓M↓) 26% 15	0.269 (P↓L↑) 41% 6
	wt14	0.385 (P↑D↑L↑M↑) 67% 1	0.352 (D↑L↑M↑) 52% 1	0.297 (P↓l↑) 29% 7	0.275 (P↓D↓m↓) 19% 11	0.302 (P↓L↑) 31% 5

Table 5.2: nDCG@20 on TREC Web Track 2012–14 when re-ranking search results from *QL*. The comparisons are conducted between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach. In addition, the relative improvements (%) and ranks among all runs within the respective years according to nDCG@20 are also reported after the absolute scores.

search results from the query-likelihood model. The ranks of the re-ranked runs are also reported when sorting the re-ranked search results together with other competing runs from the same year according to ERR@20 or nDCG@20. The characters in brackets indicate a significant difference compared with other methods, marked using uppercase or lowercase characters, namely P/p for PACRR, D/d for DRMM, L/l for DUETL and M/m for MatchPyramid, representing the two-tailed significance at 95% or 90% confidence levels, respectively.

It can be seen that, by simply re-ranking the search results from the query-likelihood method, RE-PACRR can already achieve the best runs in all three years. Beyond that, compared with the query-likelihood results, the average improvement in terms of ERR@20 is more than 99% and the average improvement in terms of nDCG@20 is 104%.

RERANKALL. Given that the search results from *QL* only account for a small subset of all judged documents and different retrieval models return different subsets, we evaluate our re-ranker's performance when re-ranking all submitted runs from the TREC Web Track 2009–14. This evaluation focuses on two aspects: how many different runs we can improve upon and by how much we improve. The former aspect is about the adaptability of a neural IR model, investigating whether it can make improvements based on different kinds of retrieval models, while the latter aspect focuses on the magnitude of improvements. Tables 5.3 and 5.4 summarize the percentages of systems that see improvements based on either ERR@20 and nDCG@20 out of the total number of systems in each year. In Tables 5.5 and 5.6, we further report the average percentage of improvements.

Tables 5.3 and 5.4 demonstrate that on average more than 95% of runs are improved by the RE-PACRR in terms of both measures. When compared with other models, RE-PACRR significantly outperforms all baseline models on five years out of six in terms of ERR@20. It is only significantly better than the results from PACRR in 2013 and 14 when measured with nDCG@20, but outperforms the other baselines in all six years. Note that, compared with nDCG@20, ERR@20 emphasizes the quality of the top-ranked documents and heavily penalizes relevant documents that are ranked lower by a model when enough relevant documents have been observed earlier (Chapelle et al., 2009). This means that the improvement of the ERR for a model mainly comes from improvements on queries for which search results at the top are not good enough from an initial ranker, and hardly from the queries where an initial ranker already performs well and a re-ranker makes the search results even better by adjusting the positions of documents in lower positions. This could explain the differences between results in terms of ERR@20 and nDCG@20, indicating that the two measures actually reflect different types of improvements. Furthermore, in Tables 5.5 and 5.6, the average improvements on different runs are summarized, and on average the initial runs get improved by 54% and 67% of ERR@20 and nDCG@20, respectively. Under both measures, RE-PACRR performs significantly better than all baselines in five years out of six years.

Measures	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
ERR@20	wt09	91% (D↑L↑)	92% (D↑L↑m↑)	86% (p↓D↑l↑)	77% (P↓m↓)	72% (P↓M↓)
	wt10	98% (P↑D↑L↑M↑)	95% (D↑L↑)	95% (D↑L↑)	69% (P↓D↓M↓)	91% (P↓L↑M↓)
	wt11	98% (P↑D↑L↑M↑)	69% (D↑L↑M↑)	43% (P↓L↑)	26% (P↓D↓M↓)	49% (P↓L↑)
	wt12	98% (P↑d↑L↑M↑)	92% (L↑)	93% (L↑)	68% (P↓D↓M↓)	95% (L↑)
	wt13	94% (P↑D↑L↑M↑)	85% (L↑M↑)	64% (P↓d↓)	61% (P↓D↓)	83% (L↑m↑)
	wt14	96% (P↑D↑L↑M↑)	84% (L↑M↑)	58% (P↓)	52% (P↓)	68%

Table 5.3: The percentage of runs that show improvements in terms of a measure when re-ranking all runs from the TREC Web Track 2009–14 based on Err@20. The comparisons are conducted in between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach.

Measures	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
nDCG@20	wt09	94% (D↑L↑M↑)	96% (D↑L↑M↑)	82% (P↓D↑L↑)	71% (P↓M↓)	68% (P↓M↓)
	wt10	98% (D↑L↑M↑)	98% (D↑L↑m↑)	95% (p↓L↑)	72% (P↓D↓M↓)	94% (P↓L↑)
	wt11	98% (D↑L↑M↑)	94% (D↑L↑M↑)	49% (P↓L↑)	31% (P↓D↓M↓)	56% (P↓L↑)
	wt12	98% (D↑L↑M↑)	97% (D↑L↑M↑)	90% (P↓L↑)	68% (P↓D↓M↓)	92% (P↓L↑)
	wt13	91% (P↑D↑L↑M↑)	88% (D↑L↑M↑)	80% (P↓L↑)	65% (P↓D↓M↓)	80% (P↓L↑)
	wt14	97% (P↑D↑L↑M↑)	90% (D↑L↑M↑)	69% (P↓l↑)	59% (P↓D↓m↓)	74% (P↓L↑)

Table 5.4: The percentage of runs that show improvements in terms of a measure when re-ranking all runs from the TREC Web Track 2009–14 based on nDCG@20. The comparisons are conducted in between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach.

PairAccuracy. Ideally, a re-ranking model should make correct decisions when ranking all documents. Therefore, we further rely on a pairwise ranking task to compare different models in this regard. In particular, given a query and a set of documents, different models assign a score to each document according to their inferred relevance relative to the given query. Thereafter, all pairs of documents are examined and the pairs that are ranked in concordance with the ground-truth judgments from TREC are deemed correct, based on which an aggregated accuracy is reported on all such document pairs in different years. For example, given query q and two documents d_1 and d_2, along with their ground-truth judgments $label(d_1)$ and $label(d_2)$, a re-ranking model provides their relevance scores as $rel(q, d_1)$ and $rel(q, d_2)$. The re-ranking model is correct when it predicts these two documents in the same order as in the ranking from the ground-truth label, e.g., $rel(q, d_1) > rel(q, d_2)$ and $label(d_1) > label(d_2)$. The relevance judgments in the TREC Web Track include up to six relevance levels: junk pages (Junk), non-relevant (NRel), relevant

Measures	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
ERR@20	wt09	43% (D↑L↑M↑)	40% (D↑L↑M↑)	31% (P↓D↑L↑)	22% (P↓M↓)	20% (P↓M↓)
	wt10	98% (P↑D↑L↑M↑)	74% (D↑L↑M↑)	54% (P↓d↑L↑)	23% (P↓M↓)	44% (P↓m↓)
	wt11	33% (P↑D↑L↑M↑)	11% (D↑L↑M↑)	-4% (P↓)	-11% (P↓D↓)	-0% (P↓L↑)
	wt12	89% (P↑D↑L↑)	66% (L↑)	68% (L↑)	22% (P↓D↓M↓)	70% (L↑)
	wt13	36% (P↑D↑L↑M↑)	27% (L↑M↑)	9% (P↓D↓)	8% (P↓D↓)	20% (L↑M↑)
	wt14	29% (P↑D↑L↑M↑)	16% (d↑L↑M↑)	5% (P↓)	2% (P↓)	8% (p↓)

Table 5.5: The average differences of the measure score for individual runs when re-ranking all runs from the TREC Web Track 2009–14 based on ERR@20. The comparisons are conducted between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach.

Measures	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
nDCG@20	wt09	36% (D↑L↑M↑)	35% (D↑L↑M↑)	20% (P↓D↑)	14% (P↓)	12% (P↓M↓)
	wt10	125% (P↑D↑L↑M↑)	96% (D↑L↑M↑)	66% (P↓L↑)	31% (P↓M↓)	58% (P↓)
	wt11	43% (P↑D↑L↑M↑)	26% (D↑L↑M↑)	0% (P↓d↓l↑)	-11% (P↓D↓m↓)	7% (P↓L↑m↑)
	wt12	122% (P↑D↑L↑M↑)	89% (D↑L↑m↑)	65% (p↓L↑)	21% (P↓D↓M↓)	69% (P↓L↑)
	wt13	43% (P↑D↑L↑M↑)	31% (D↑L↑M↑)	16% (P↓L↑)	9% (P↓d↓M↓)	15% (P↓l↑)
	wt14	31% (P↑D↑L↑M↑)	21% (D↑L↑M↑)	12% (P↓l↑)	6% (P↓D↓m↓)	11% (P↓L↑)

Table 5.6: The average differences of the measure score for individual runs when re-ranking all runs from the TREC Web Track 2009–14 based on nDCG@20. The comparisons are conducted between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach.

(Rel), highly relevant (HRel), key pages (Key), and navigational pages (Nav), corresponding to six graded levels, i.e., -2, 0, 1, 2, 3, 4. As suggested in (Hui et al., 2017b), a navigational document is different from other relevant documents in terms of the user intent it satisfies, where links to other relevant documents for the same query are provided. Thus, documents labeled with Nav are not considered in this task. Moreover, documents labeled as Junk and NRel, and documents labeled as HRel and Key are merged into NRel and HRel, respectively, due to the limited number of documents labeled as Junk and Key. Then, all pairs of documents with different labels are generated as test pairs. In total, these three label pairs account for 95% of all document pairs according to the column named "volume" in Tables 5.7, 5.8 and 5.9.

The results show that significant improvements relative to all baselines are observed in three out of all six years among the three label pairs. Compared with DUETL, MatchPyramid, and

Label Pair	volume (%)	# queries	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
HRel-NRel	23.1%	262	wt09	0.715 (P↑D↑L↑M↑)	0.694 (D↑L↑M↑)	0.656 (P↓D↑l↑)	0.632 (P↓D↑m↓)	0.602 (P↓L↓M↓)
			wt10	0.846 (D↑L↑M↑)	0.828 (D↑L↑M↑)	0.777 (P↓D↑L↑)	0.707 (P↓M↓)	0.739 (P↓M↓)
			wt11	0.837 (P↑D↑L↑M↑)	0.789 (D↑L↑M↑)	0.757 (P↓D↑L↑)	0.700 (P↓d↓M↓)	0.735 (P↓l↑M↓)
			wt12	0.826 (P↑D↑L↑M↑)	0.795 (D↑L↑M↑)	0.741 (P↓D↑L↑)	0.655 (P↓d↓M↓)	0.694 (P↓l↑M↓)
			wt13	0.758 (D↑L↑M↑)	0.749 (D↑L↑M↑)	0.691 (P↓D↑L↑)	0.647 (P↓M↓)	0.637 (P↓M↓)
			wt14	0.766 (D↑L↑M↑)	0.772 (D↑L↑M↑)	0.672 (P↓d↑)	0.648 (P↓)	0.649 (P↓m↓)

Table 5.7: Comparison among tested methods in terms of accuracy in ranking document pairs with *HRel-NRel*. The columns "volume" and "# queries" record the occurrences of each label combination out of the total pairs, and the number of queries that include a particular label combination among all six years, respectively. The comparisons are conducted between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approaches.

Label Pair	volume (%)	# queries	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
			wt09	0.531	0.539	0.538	0.534	0.539
			wt10	0.587 (p↑D↑L↑)	0.571 (D↑L↑)	0.581 (D↑L↑)	0.529 (P↓M↓)	0.544 (P↓M↓)
			wt11	0.582 (P↑d↓L↑)	0.537 (D↓M↓)	0.608 (P↑L↑)	0.536 (D↓M↓)	0.606 (P↑L↑)
HRel-Rel	8.4%	257	wt12	0.671 (D↑L↑M↑)	0.655 (D↑L↑M↑)	0.607 (P↓D↑L↑)	0.550 (P↓M↓)	0.549 (P↓M↓)
			wt13	0.572 (D↑L↑)	0.573 (d↑L↑)	0.564 (l↑)	0.545 (P↓m↓)	0.552 (p↓)
			wt14	0.602 (P↑D↑L↑M↑)	0.581 (D↑)	0.575 (D↑)	0.551	0.544 (P↓M↓)

Table 5.8: Comparison among tested methods in terms of accuracy in ranking document pairs with *HRel-Rel*. The columns "volume" and "# queries" record the occurrences of each label combination out of the total pairs, and the number of queries that include a particular label combination among all six years, respectively. The comparisons are conducted between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approaches.

Label Pair	volume (%)	# queries	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
Rel-NRel	63.5%	290	wt09	0.682 (P↑D↑L↑M↑)	0.660 (D↑L↑M↑)	0.619 (P↓D↑)	0.598 (P↓D↑)	0.565 (P↓L↓M↓)
			wt10	0.799 (D↑L↑M↑)	0.788 (D↑L↑M↑)	0.718 (P↓L↑)	0.686 (P↓M↓)	0.708 (P↓)
			wt11	0.782 (D↑L↑M↑)	0.771 (D↑L↑M↑)	0.651 (P↓D↑)	0.650 (P↓D↑)	0.614 (P↓L↓M↓)
			wt12	0.741 (P↑D↑L↑M↑)	0.725 (D↑L↑M↑)	0.667 (P↓L↑)	0.619 (P↓D↓M↓)	0.658 (P↓L↑)
			wt13	0.707 (p↑D↑L↑M↑)	0.692 (D↑L↑M↑)	0.635 (P↓D↑L↑)	0.605 (P↓d↑M↓)	0.586 (P↓l↓M↓)
			wt14	0.700 (P↓D↑L↑M↑)	0.721 (D↑L↑M↑)	0.612 (P↓)	0.597 (P↓)	0.603 (P↓)

Table 5.9: Comparison among tested methods in terms of accuracy in ranking document pairs with *Rel-NRel*. The columns “volume” and “# queries” record the occurrences of each label combination out of the total pairs, and the number of queries that include a particular label combination among all six years, respectively. The comparisons are conducted between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in brackets indicate a significant difference under two-tailed paired Student’s t-tests at 95% or 90% confidence levels relative to the corresponding approaches.

DRMM, RE-PACRR performs better in all six years for labeling pairs between HRel and NRel, as well as between Rel and NRel. As for document pairs that are labeled as HRel and Rel, the RE-PACRR performs relatively close to the baselines. In terms of absolute accuracy, on average, RE-PACRR yields correct predictions on 79.1%, 73.5%, and 59.1% of document pairs for label pairs HRel–NRel, Rel–NRel, and HRel–Rel, respectively, where the decreasing accuracy is due to the increasing difficulty on these document pairs.

5.4.3 Ablation Test

In this section, we add individual components to PACRR to examine the effects of different building blocks. The comparison is primarily relative to the results from the PACRR model. Intuitively, after combining a component with PACRR model, if the new model outperforms the results from PACRR, one could argue that the component actually contributes to the performance of the RE-PACRR model. As aforementioned, the significant difference relative to PACRR is summarized in terms of P/p following the reported numbers.

The comparison in terms of RERANKSIMPLE benchmark is firstly summarized in Tables 5.10 and 5.11. It can be seen that the permutation component leads to significant improvements over WT12–14 consistently in terms of nDCG@20, whereas proximity hurt the performance on at least one year in terms of either measure. As for the other three components, the context component only leads to significant improvements on WT14, meanwhile there is no clear conclusion for disambiguation and cascade building blocks.

We further summarize results in terms of RERANKALL in Tables 5.12, 5.13, 5.14 and 5.15, indicating the percentage of improved runs and the average improvement among all runs respectively. It can be seen that both permutation and context components lead to consistent improvements on at least three years in terms of nDCG@20. Meanwhile, the effects from other three building blocks are unclear, namely, the comparison of performance is not consist among different years.

Finally, the methods are compared in terms of PAIRACCURACY as in Tables 5.16, 5.17 and 5.18. Similar to the observations from the other two benchmarks, the permutation building block leads to consistently significant improvement over PACRR. Meanwhile, one could observe improvements from the column corresponding to context component over half of the six years. Differently, it seems cascade component also leads to improvements on more than half of the six years, but hurts the performance WT13 when distinguishing *HRel* and *Rel*. There is no clear conclusion for the disambiguation and the proximity components.

Measures	Year	-vocabulary expansion		+cascade		+context		+proximity		+permutation	
	wt12	0.356 (p↓D↑L↑M↑)	101% 1	0.342 (P↓N↓d↑L↑M↑)	93% 1	0.371 (D↑L↑M↑)	110% 1	0.321 (P↓N↓L↑)	81% 3	0.379 (D↑L↑M↑)	114% 1
ERR@20	wt13	0.167 (L↑M↑)	66% 5	0.178 (D↑L↑M↑)	77% 3	0.183 (D↑L↑M↑)	82% 2	0.156 (P↓N↓L↑M↑)	55% 5	0.167 (L↑M↑)	66% 4
	wt14	0.226 (N↓D↑L↑M↑)	72% 1	0.246 (P↑D↑L↑M↑)	88% 1	0.245 (P↑D↑L↑M↑)	87% 1	0.214 (N↓d↑L↑M↑)	63% 4	0.237 (D↑L↑M↑)	81% 1

Table 5.10: Err@20 on TREC Web Track 2012–14 when re-ranking search results from *QL*. The comparisons are conducted in between the model adding one component to PACRR and neu-PACRR (N/n), PACRR (P/p), DRMM (D/d), Duet-local (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach. In addition, the relative improvements (%) and ranks among all runs within the respective years according to Err@20 are also reported directly after the absolute scores.

Measures	Year	-vocabulary expansion		+cascade		+context		+proximity		+permutation	
	wt12	0.258 (D↑L↑M↑)	143% 1	0.254 (N↓D↑L↑M↑)	139% 1	0.255 (n↓D↑L↑M↑)	140% 1	0.231 (p↓N↓d↑L↑)	117% 3	0.277 (P↑D↑L↑M↑)	160% 1
nDCG@20	wt13	0.308 (D↑L↑M↑)	62% 2	0.323 (D↑L↑M↑)	70% 2	0.331 (D↑L↑M↑)	74% 1	0.297 (d↑L↑M↑)	56% 3	0.331 (p↑D↑L↑M↑)	74% 2
	wt14	0.350 (p↑N↓D↑L↑M↑)	52% 1	0.370 (P↑D↑L↑M↑)	60% 1	0.371 (P↑N↓D↑L↑M↑)	61% 1	0.333 (N↓D↑L↑M↑)	44% 2	0.361 (P↑n↓D↑L↑M↑)	56% 1

Table 5.11: nDCG@20 on TREC Web Track 2012–14 when re-ranking search results from *QL*. The comparisons are conducted in between the model adding one component to PACRR and neu-PACRR (N/n), PACRR (P/p), DRMM (D/d), Duet-local (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach. In addition, the relative improvements (%) and ranks among all runs within the respective years according to nDCG@20 are also reported directly after the absolute scores.

Measures	Year	-vocabulary expansion	+cascade	+context	+proximity	+permutation
ERR@20	wt09	40% (n↓D↑L↑M↑)	40% (N↓D↑L↑M↑)	41% (N↓D↑L↑M↑)	36% (N↓D↑L↑m↑)	42% (P↑N↓D↑L↑M↑)
	wt10	74% (N↓D↑L↑M↑)	90% (P↑N↓D↑L↑M↑)	88% (p↑N↓D↑L↑M↑)	72% (N↓D↑L↑M↑)	78% (N↓D↑L↑M↑)
	wt11	15% (n↓D↑L↑M↑)	16% (n↓D↑L↑M↑)	33% (P↑D↑L↑M↑)	9% (N↓D↑L↑M↑)	34% (P↑D↑L↑M↑)
	wt12	76% (L↑)	74% (L↑)	73% (n↓L↑)	64% (P↓N↓L↑)	86% (D↑L↑)
	wt13	22% (n↓L↑M↑)	26% (L↑M↑)	36% (p↑D↑L↑M↑)	23% (L↑M↑)	25% (L↑M↑)
	wt14	18% (N↓D↑L↑M↑)	27% (P↑D↑L↑M↑)	32% (P↑D↑L↑M↑)	14% (N↓L↑M↑)	24% (p↑D↑L↑M↑)

Table 5.12: The average statistics when re-ranking all runs from the TREC Web Track 2009–14 based on Err@20. The percentage of runs that show improvements in terms of a measure is summarized. The comparisons are conducted in between the model adding one component to PACRR and neu-PACRR (N/n), PACRR (P/p), DRMM (D/d), Duet-local (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach. For example, in wt12 the PACRR-firstk method significantly outperforms all three baselines at 95% confidence levels when using both measures.

Measures	Year	-vocabulary expansion	+cascade	+context	+proximity	+permutation
nDCG@20	wt09	34% (P↑D↑L↑M↑)	35% (P↑D↑L↑M↑)	37% (P↑D↑L↑M↑)	28% (N↓D↑L↑M↑)	37% (P↑D↑L↑M↑)
	wt10	95% (N↓D↑L↑M↑)	114% (N↓D↑L↑M↑)	115% (N↓D↑L↑M↑)	94% (N↓D↑L↑M↑)	106% (N↓D↑L↑M↑)
	wt11	27% (N↓D↑L↑M↑)	32% (D↑L↑M↑)	46% (P↑D↑L↑M↑)	26% (n↓D↑L↑M↑)	45% (P↑D↑L↑M↑)
	wt12	97% (N↓D↑L↑M↑)	103% (P↑D↑L↑M↑)	110% (P↑D↑L↑M↑)	84% (N↓d↑L↑)	116% (P↑D↑L↑M↑)
	wt13	27% (D↑L↑M↑)	32% (D↑L↑M↑)	37% (p↑D↑L↑M↑)	27% (D↑L↑M↑)	35% (p↑D↑L↑M↑)
	wt14	20% (p↑N↓D↑L↑M↑)	24% (P↑N↓D↑L↑M↑)	30% (P↑D↑L↑M↑)	15% (N↓L↑)	24% (P↑N↓D↑L↑M↑)

Table 5.13: The average statistics when re-ranking all runs from the TREC Web Track 2009–14 based on nDCG@20. The percentage of runs that show improvements in terms of a measure is summarized. The comparisons are conducted in between the model adding one component to PACRR and neu-PACRR (N/n), PACRR (P/p), DRMM (D/d), Duet-local (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach. For example, in wt12 the PACRR-firstk method significantly outperforms all three baselines at 95% confidence levels when using both measures.

5.5 Discussion

In this chapter, we have demonstrated the importance of preserving positional information for deep retrieval models by incorporating insights from traditional information retrieval models into our PACRR novel deep retrieval model. In particular, PACRR captures term dependencies and proximity among query term occurrences through the use of convolutional layers that consider document and query n-grams of different sizes. Our model considers document relevance across different query terms through the use of several feedforward layers that combines relevance signals across query terms. On top of that, a novel neural IR model named RE-PACRR is proposed, better incorporating a number of task-specific insights from traditional ad-hoc retrieval, and boosting PACRR with several novel building blocks. Extensive experiments on TREC Web Track data confirm that both PACRR and RE-PACRR perform better than all baseline models on three

Measures	Year	-vocabulary expansion	+cascade	+context	+proximity	+permutation
ERR@20	wt09	93% (D↑L↑)	94% (p↑D↑L↑m↑)	90% (N↓D↑L↑)	92% (D↑L↑)	88% (P↓N↓D↑L↑)
	wt10	95% (N↓D↑L↑)	97% (D↑L↑m↑)	97% (D↑L↑m↑)	95% (N↓D↑L↑)	96% (N↓D↑L↑)
	wt11	73% (N↓D↑L↑M↑)	85% (D↑L↑M↑)	94% (p↑D↑L↑M↑)	68% (N↓D↑L↑M↑)	96% (P↑D↑L↑M↑)
	wt12	94% (N↓L↑)	94% (L↑)	97% (L↑M↑)	90% (P↓N↓d↓L↑)	97% (L↑M↑)
	wt13	79% (L↑m↑)	89% (L↑M↑)	89% (L↑M↑)	85% (L↑M↑)	86% (L↑M↑)
	wt14	86% (N↓d↑L↑M↑)	90% (p↑n↓D↑L↑M↑)	96% (P↑D↑L↑M↑)	76% (N↓L↑m↑)	93% (P↑D↑L↑M↑)

Table 5.14: The average statistics when re-ranking all runs from the TREC Web Track 2009–14 based on ERR@20. The average differences of the measure score for individual runs are reported The comparisons are conducted in between the model adding one component to PACRR and neu-PACRR (N/n), PACRR (P/p), DRMM (D/d), Duet-local (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach.

Measures	Year	-vocabulary expansion	+cascade	+context	+proximity	+permutation
nDCG@20	wt09	95% (D↑L↑M↑)	96% (D↑L↑M↑)	96% (p↑D↑L↑M↑)	91% (N↓D↑L↑M↑)	96% (p↑D↑L↑M↑)
	wt10	98% (D↑L↑m↑)	98% (D↑L↑M↑)	98% (D↑L↑M↑)	98% (D↑L↑M↑)	98% (D↑L↑M↑)
	wt11	95% (D↑L↑M↑)	96% (N↓D↑L↑M↑)	98% (p↑D↑L↑M↑)	88% (D↑L↑M↑)	98% (p↑D↑L↑M↑)
	wt12	98% (D↑L↑M↑)	98% (D↑L↑M↑)	98% (D↑L↑M↑)	96% (N↓d↑L↑m↑)	98% (D↑L↑M↑)
	wt13	86% (d↑L↑m↑)	89% (D↑L↑M↑)	89% (D↑L↑M↑)	86% (L↑m↑)	90% (P↑D↑L↑M↑)
	wt14	90% (p↑N↓D↑L↑M↑)	89% (p↑N↓D↑L↑M↑)	96% (P↑D↑L↑M↑)	80% (N↓L↑)	94% (P↑D↑L↑M↑)

Table 5.15: The average statistics when re-ranking all runs from the TREC Web Track 2009–14 based on nDCG@20. The average differences of the measure score for individual runs are reported The comparisons are conducted in between the model adding one component to PACRR and neu-PACRR (N/n), PACRR (P/p), DRMM (D/d), Duet-local (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach.

benchmarks. Finally, we conduct an ablation study to better understand the contribution of the individual novel components as well as their interplay. As for future work, one could further exploit insights from the literature on ad-hoc retrieval and to devise suitable components to make them accessible to a neural model.

Measures	volume (%)	# queries	Year	-vocabulary expansion	+cascade	+context	+proximity	+permutation
HRel-NRel	23.1%	262	wt09	0.692 (N↓D↑L↑M↑)	0.699 (N↓D↑L↑M↑)	0.701 (N↓D↑L↑M↑)	0.683 (N↓D↑L↑M↑)	0.705 (p↑N↓D↑L↑M↑)
			wt10	0.829 (N↓D↑L↑M↑)	0.834 (D↑L↑M↑)	0.841 (p↑D↑L↑M↑)	0.825 (N↓D↑L↑M↑)	0.844 (p↑D↑L↑M↑)
			wt11	0.793 (N↓D↑L↑M↑)	0.797 (p↑N↓D↑L↑M↑)	0.828 (P↑D↑L↑M↑)	0.809 (P↑N↓D↑L↑M↑)	0.827 (P↑n↓D↑L↑M↑)
			wt12	0.795 (P↑n↓D↑L↑M↑)	0.803 (P↑D↑L↑M↑)	0.780 (P↑N↓D↑L↑M↑)	0.797 (P↑D↑L↑M↑)	0.814 (P↑D↑L↑M↑)
			wt13	0.742 (P↑D↑L↑M↑)	0.746 (P↑D↑L↑M↑)	0.749 (P↑D↑L↑M↑)	0.725 (D↑L↑M↑)	0.757 (P↑n↑D↑L↑M↑)
			wt14	0.764 (P↑D↑L↑M↑)	0.770 (P↑D↑L↑M↑)	0.772 (P↑D↑L↑M↑)	0.737 (N↓D↑L↑M↑)	0.780 (P↑D↑L↑M↑)

Table 5.16: Comparison among tested methods in terms of accuracy in ranking document pairs with *HRel-NRel*. The column named “volume” and # queries records the occurrences of each label combination out of the total pairs among, and the number of queries that include a particular label combination among all six years respectively. The comparisons are conducted in between the model adding one component to PACRR and neu-PACRR (N/n), PACRR (P/p), DRMM (D/d), Duet-local (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in the brackets indicate a significant difference under two-tailed paired Student’s t-tests at 95% or 90% confidence levels relative to the corresponding approach.

Measures	volume (%)	# queries	Year	-vocabulary expansion	+cascade	+context	+proximity	+permutation
HRel-Rel	8.4%	257	wt09	0.538	0.533 (n↓)	0.545 (m↑)	0.512 (P↓N↓D↓L↓M↓)	0.543
			wt10	0.575 (n↓D↑L↑)	0.585 (P↑D↑L↑)	0.571 (N↓D↑L↑)	0.578 (n↓D↑L↑)	0.582 (p↑D↑L↑)
			wt11	0.533 (N↓D↓M↓)	0.530 (N↓D↓M↓)	0.568 (P↑D↓l↑M↓)	0.563 (P↑D↓M↓)	0.562 (P↑N↓D↓M↓)
			wt12	0.651 (p↑D↑L↑M↑)	0.671 (P↑D↑L↑M↑)	0.646 (P↑D↑L↑M↑)	0.653 (p↑D↑L↑M↑)	0.659 (P↑D↑L↑M↑)
			wt13	0.562	0.552 (P↓)	0.584 (N↑D↑L↑)	0.560 (l↑)	0.587 (P↑N↑D↑L↑M↑)
			wt14	0.573 (P↑N↓D↑)	0.584 (P↑D↑)	0.599 (P↑D↑L↑M↑)	0.570 (P↑N↓D↑)	0.584 (P↑D↑)

Table 5.17: Comparison among tested methods in terms of accuracy in ranking document pairs with *HRel-Rel*. The column named "volume" and # queries records the occurrences of each label combination out of the total pairs among, and the number of queries that include a particular label combination among all six years respectively. The comparisons are conducted in between the model adding one component to PACRR and neu-PACRR (N/n), PACRR (P/p), DRMM (D/d), Duet-local (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach.

Measures	volume (%)	# queries	Year	-vocabulary expansion	+cascade	+context	+proximity	+permutation
			wt09	0.658 (N↓D↑L↑M↑)	0.669 (p↑D↑L↑M↑)	0.663 (N↓D↑L↑M↑)	0.668 (D↑L↑M↑)	0.667 (p↑n↓D↑L↑M↑)
			wt10	0.785 (N↓D↑L↑M↑)	0.787 (D↑L↑M↑)	0.803 (p↑D↑L↑M↑)	0.781 (N↓D↑L↑M↑)	0.804 (P↑D↑L↑M↑)
Rel-NRel	63.5%	290	wt11	0.774 (p↑D↑L↑M↑)	0.773 (D↑L↑M↑)	0.780 (P↑D↑L↑M↑)	0.761 (D↑L↑M↑)	0.791 (P↑n↑D↑L↑M↑)
			wt12	0.728 (P↑D↑L↑M↑)	0.722 (P↑N↓D↑L↑M↑)	0.727 (P↑D↑L↑M↑)	0.716 (D↑L↑M↑)	0.739 (P↑D↑L↑M↑)
			wt13	0.691 (P↑D↑L↑M↑)	0.703 (P↑D↑L↑M↑)	0.686 (P↑n↓D↑L↑M↑)	0.684 (n↓D↑L↑M↑)	0.695 (P↑D↑L↑M↑)
			wt14	0.716 (P↑D↑L↑M↑)	0.707 (p↑D↑L↑M↑)	0.705 (D↑L↑M↑)	0.689 (n↓D↑L↑M↑)	0.721 (P↑n↑D↑L↑M↑)

Table 5.18: Comparison among tested methods in terms of accuracy in ranking document pairs with *Rel-NRel*. The column named "volume" and # queries records the occurrences of each label combination out of the total pairs among, and the number of queries that include a particular label combination among all six years respectively. The comparisons are conducted in between the model adding one component to PACRR and neu-PACRR (N/n), PACRR (P/p), DRMM (D/d), Duet-local (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach.

6 Conclusion

In this thesis, we resolved two difficulties in information retrieval, namely, the *large number of query-document pairs* in manual judgments for IR evaluation and the *complicated patterns* to be considered when building a retrieval model.

As the first contribution, in Chapter 3, we first examined different document representations in accordance with the *cluster hypothesis* (Hui and Berberich, 2016). Thereafter, we investigated how different strategies for selective labeling and mitigating incomplete labels interact, and accordingly proposed a novel strategy MaxRep (Hui and Berberich, 2015). In contrast to existing strategies, it considers both ranking information and document contents and seeks to select a representative subset of documents to label. Our experiments confirmed that MaxRep outperforms other rivaling strategies. In addition, as for the evaluation in novelty and diversity, we proposed a cascade framework named Lmd-Cascade (Hui et al., 2017a), which *mitigates incomplete labels* by employing Kullback-Leibler divergence between language models estimated for the subtopics and for the search results. The experiments demonstrated that Lmd-Cascade is robust as an evaluation measure even when as little as 15% of relevance judgments are available.

As the second contribution, in Chapter 4, we first demonstrated that ties among preference judgments can dramatically reduce the number of judgments (Hui and Berberich, 2017a), based on which we proposed a novel judgment mechanism, coined Merge-Tie-Judge (Hui and Berberich, 2017b), which is more robust and can utilize the "cluster effect" to reduce judgments. Through empirical study, we showed that the preference judgments with and without ties behave quite differently in terms of transitivity, time consumption and the judgment quality, but both can lead to better judgments than the graded judgments when collecting via crowdsourcing, whereas only the one from strict preference judgments is comparable to the manual judgments from Trec (Hui and Berberich, 2017c). Ultimately, we pointed out that ties can be exploited as a measure of comprimise between the cost and the quality of judgments.

As the third contribution, in Chapter 5, we highlighted the importance of preserving positional information for deep retrieval models (Hui et al., 2017c) by incorporating insights from the literature, and proposed neural IR models, named *PACRR* (Hui et al., 2017b) and *RE-PACRR*[1] (Hui

[1] https://arxiv.org/abs/1706.10192v2

et al., 2018). PACRR captures term dependencies and proximity among query term occurrences through the use of convolutional layers that consider n-gram matches of different sizes. On top of that, RE-PACRR better incorporates a number of task-specific insights from traditional ad-hoc retrieval, and boosts PACRR with several novel building blocks. Intensive experiments on TREC Web Track data confirmed that both PACRR and RE-PACRR perform better than all baseline models on three benchmarks, and RE-PACRR is actually the best-performed neural IR model at the time of writing. In the end, we also conducted an ablation study to better understand the contributions of the individual novel components as well as their interplay.

6.1 Outlook

As future works, novel supervised/unsupervised learning methods to mitigate the missing labels, especially the ones incorporating deep learning, are highly desired, given that the manual judgments are always necessary for evaluation and for training. In particular, novel selective labeling methods should be designed to work together with follow-up deep prediction models, which are very different from the traditional prediction models investigated in Chapter 3.

Moreover, the assumption about the transitivity should be softened to make the proposed MERGE-TIE-JUDGE more applicable in practice, possibly through the development of novel judgment mechanisms which can tolerate intransitive judgments. In addition, to make the preference judgments more practical to use, ultimately replacing the graded judgments, the special designs of crowdsourcing task are desired to incorporate the fact that different document pairs are not independent with each other. Put differently, given that the huge amount of document pairs are actually derived based on several few queries, workload should be assigned such that document pairs that are about the same or similar topics should be assigned to the same assessors, taking advantages of their familiarity of the particular topics.

Finally, as for the neural IR models, one could further exploit insights from the literature on ad-hoc retrieval and devise suitable components to make them accessible to a neural IR model. Moreover, the established neural retrieval models are expensive to use, therefore improving the efficiency of the neural IR model would be another interesting research direction. In addition, benchmarks for the evaluation of the neural IR models are of interests given that neural IR models are mostly re-rankers, making the evaluation different from the one in traditional IR. More importantly, given the fact that training a neural IR model could be very data hungry, a large enough labeled dataset is desirable to make different neural IR models comparable by training them on the same data.

Bibliography

Agrawal, Rakesh, Gollapudi, Sreenivas, Halverson, Alan, and Ieong, Samuel (2009). “Diversifying search results.” In: *Proceedings of the Second ACM International Conference on Web Search and Data Mining*. WSDM ’09. Barcelona, Spain: ACM, pp. 5–14. ISBN: 978-1-60558-390-7 (cit. on p. 13).

Alonso, Omar and Baeza-Yates, Ricardo (2011). “Design and implementation of relevance assessments using crowdsourcing.” In: *European Conference on Information Retrieval*. ECIR ’11. Springer, pp. 153–164 (cit. on pp. 11, 57, 70).

Alonso, Omar and Mizzaro, Stefano (2009). “Can we get rid of TREC assessors? Using Mechanical Turk for relevance assessment.” In: *Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation*. Vol. 15, p. 16 (cit. on pp. 2, 54, 57, 76).

Alonso, Omar and Mizzaro, Stefano (2012). “Using crowdsourcing for TREC relevance assessment.” In: *Information processing & management* 48.6, pp. 1053–1066 (cit. on pp. 10, 57, 71, 73–76).

Aslam, Javed A., Pavlu, Virgil, and Yilmaz, Emine (2006). “A Statistical Method for System Evaluation Using Incomplete Judgments.” In: *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’06. Seattle, Washington, USA: ACM, pp. 541–548. ISBN: 1-59593-369-7 (cit. on p. 24).

Bashir, Maryam, Anderton, Jesse, Wu, Jie, Golbus, Peter B, Pavlu, Virgil, and Aslam, Javed A (2013). “A document rating system for preference judgements.” In: *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. SIGIR ’13. ACM, pp. 909–912 (cit. on pp. 54, 56).

Blei, David M, Ng, Andrew Y, and Jordan, Michael I (2003). “Latent dirichlet allocation.” In: *Journal of machine Learning research* 3.Jan, pp. 993–1022 (cit. on pp. 16, 23, 27).

Bompada, Tanuja, Chang, Chi-Chao, Chen, John, Kumar, Ravi, and Shenoy, Rajesh (2007). “On the Robustness of Relevance Measures with Incomplete Judgments.” In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’07. Amsterdam, The Netherlands: ACM, pp. 359–366. ISBN: 978-1-59593-597-7 (cit. on p. 41).

Buckley, Chris and Voorhees, Ellen M. (2004). “Retrieval Evaluation with Incomplete Information.” In: *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’04. Sheffield, United Kingdom: ACM, pp. 25–32. ISBN: 1-58113-881-4 (cit. on pp. 2, 22, 24, 25, 33, 34, 41).

Büttcher, Stefan, Clarke, Charles L. A., Yeung, Peter C. K., and Soboroff, Ian (2007). "Reliable information retrieval evaluation with incomplete and biased judgements." In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '07, pp. 63–70 (cit. on pp. 16, 22, 23, 25, 30, 34).

Cao, Yunbo, Xu, Jun, Liu, Tie-Yan, Li, Hang, Huang, Yalou, and Hon, Hsiao-Wuen (2006). "Adapting ranking SVM to document retrieval." In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '06. ACM, pp. 186–193 (cit. on p. 61).

Carbonell, Jaime and Goldstein, Jade (1998). "The use of MMR, diversity-based reranking for reordering documents and producing summaries." In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '98. Melbourne, Australia: ACM, pp. 335–336. ISBN: 1-58113-015-5 (cit. on p. 8).

Carterette, Ben (2007). "Robust test collections for retrieval evaluation." In: *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '07. ACM, pp. 55–62 (cit. on p. 33).

Carterette, Ben (2011). "An Analysis of NP-completeness in Novelty and Diversity Ranking." In: *Inf. Retr.* 14.1, pp. 89–106. ISSN: 1386-4564 (cit. on p. 14).

Carterette, Ben and Allan, James (2007). "Semiautomatic evaluation of retrieval systems using document similarities." In: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. CIKM '07. ACM, pp. 873–876 (cit. on pp. 25, 26).

Carterette, Ben, Allan, James, and Sitaraman, Ramesh (2006). "Minimal Test Collections for Retrieval Evaluation." In: *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '06. Seattle, Washington, USA: ACM, pp. 268–275. ISBN: 1-59593-369-7 (cit. on pp. 1, 21, 24, 33).

Carterette, Ben and Bennett, Paul N (2008). "Evaluation measures for preference judgments." In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '08. ACM, pp. 685–686 (cit. on p. 60).

Carterette, Ben, Bennett, Paul N, Chickering, David Maxwell, and Dumais, Susan T (2008). "Here or there: Preference Judgments for Relevance." In: *Advances in Information Retrieval*. Springer, pp. 16–27 (cit. on pp. 2, 53–57, 72, 74–77, 79).

Chapelle, Olivier, Ji, Shihao, Liao, Ciya, Velipasaoglu, Emre, Lai, Larry, and Wu, Su-Lin (2011). "Intent-based diversification of web search results: metrics and algorithms." English. In: *Information Retrieval* 14.6, pp. 572–592. ISSN: 1386-4564 (cit. on pp. 9, 12–14).

Chapelle, Olivier, Metzler, Donald, Zhang, Ya, and Grinspan, Pierre (2009). "Expected reciprocal rank for graded relevance." In: *Proceedings of the 18th ACM conference on Information and knowledge management*. CIKM '09. Hong Kong, China: ACM, pp. 621–630. ISBN: 978-1-60558-512-3 (cit. on pp. 12, 13, 92, 95).

Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." In: *arXiv preprint arXiv:1406.1078* (cit. on p. 20).

Clarke, Charles, Kolla, Maheedhar, Cormack, Gordon V, Vechtomova, Olga, Ashkan, Azin, Büttcher, Stefan, and MacKinnon, Ian (2008). "Novelty and diversity in information retrieval evaluation." In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '08. Singapore, Singapore: ACM, pp. 659–666. ISBN: 978-1-60558-164-4 (cit. on pp. 12, 13).

Clarke, Charles, Kolla, Maheedhar, and Vechtomova, Olga (2009). "An Effectiveness Measure for Ambiguous and Underspecified Queries." In: *Proceedings of the 2nd International Conference on Theory of Information Retrieval: Advances in Information Retrieval Theory*. ICTIR '09. Cambridge, UK: Springer-Verlag, pp. 188–199. ISBN: 978-3-642-04416-8 (cit. on pp. 12–14).

Cleverdon, Cyril W. (1991). "The significance of the Cranfield tests on index languages." In: *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '91. ACM Press, pp. 3–12 (cit. on p. 9).

Collins-Thompson, Kevyn, Bennett, Paul, Diaz, Fernando, Clarke, Charles L. A., and Voorhees, Ellen M (2015a). *TREC 2013 web track overview*. Tech. rep. DTIC Document (cit. on p. 10).

Collins-Thompson, Kevyn, Macdonald, Craig, Bennett, Paul, Diaz, Fernando, and Voorhees, Ellen M (2015b). *TREC 2014 web track overview*. Tech. rep. DTIC Document (cit. on p. 86).

Cormack, Gordon V., Palmer, Christopher R., and Clarke, Charles L. A. (1998). "Efficient Construction of Large Test Collections." In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*. SIGIR '98. ACM, pp. 282–289 (cit. on p. 23).

Cormen, Thomas H (2009). *Introduction to algorithms*. MIT press (cit. on pp. 59, 60).

Craswell, Nick, Zoeter, Onno, Taylor, Michael, and Ramsey, Bill (2008). "An experimental comparison of click position-bias models." In: *Proceedings of the 2008 International Conference on Web Search and Data Mining*. WSDM '08. ACM, pp. 87–94 (cit. on p. 84).

Cronen-Townsend, Stephen, Zhou, Yun, and Croft, W. Bruce (2004). "A framework for selective query expansion." In: *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management, Washington, DC, USA, November 8-13, 2004*. CIKM '04, pp. 236–237 (cit. on p. 82).

Dehghani, Mostafa, Zamani, Hamed, Severyn, Aliaksei, Kamps, Jaap, and Croft, W. Bruce (2017). "Neural Ranking Models with Weak Supervision." In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '17. Shinjuku, Tokyo, Japan: ACM, pp. 65–74. ISBN: 978-1-4503-5022-8 (cit. on p. 91).

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron (2016). *Deep learning*. MIT Press (cit. on pp. 2, 20, 88).

Grady, Catherine and Lease, Matthew (2010). "Crowdsourcing document relevance assessment with mechanical turk." In: *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's mechanical turk*. Association for Computational Linguistics, pp. 172–179 (cit. on p. 57).

Guo, Jiafeng, Fan, Yixing, Ai, Qingyao, and Croft, W. Bruce (2016). “A deep relevance matching model for ad-hoc retrieval.” In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. CIKM ’16. ACM, pp. 55–64 (cit. on pp. 81–83, 85, 87, 92, 93).

Gupta, Manish and Bendersky, Michael (2015). “Information Retrieval with Verbose Queries.” In: SIGIR ’15 (cit. on p. 83).

Hochreiter, Sepp and Schmidhuber, Jürgen (1997). “Long short-term memory.” In: *Neural computation* 9.8, pp. 1735–1780 (cit. on p. 20).

Howe, Jeff (2008). *Crowdsourcing: How the power of the crowd is driving the future of business*. Random House (cit. on p. 11).

Huang, Po-Sen, He, Xiaodong, Gao, Jianfeng, Deng, Li, Acero, Alex, and Heck, Larry (2013). “Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data.” In: *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*. CIKM ’13. San Francisco, California, USA: ACM, pp. 2333–2338. ISBN: 978-1-4503-2263-8 (cit. on pp. 81, 85).

Hui, Kai and Berberich, Klaus (2015). “Selective labeling and incomplete label mitigation for low-cost evaluation.” In: *International Symposium on String Processing and Information Retrieval*. SPIRE ’15. Springer International Publishing, pp. 137–148 (cit. on pp. 2, 109).

Hui, Kai and Berberich, Klaus (2016). “Cluster Hypothesis in Low-Cost IR Evaluation with Different Document Representations.” In: *Proceedings of the 25th International Conference Companion on World Wide Web*. WWW ’16. International World Wide Web Conferences Steering Committee, pp. 47–48 (cit. on pp. 2, 109).

Hui, Kai and Berberich, Klaus (2017a). “Low-Cost Preference Judgment via Ties.” In: *The 39th European Conference on Information Retrieval*. ECIR ’17. Springer International Publishing, pp. 626–632 (cit. on pp. 2, 109).

Hui, Kai and Berberich, Klaus (2017b). “Low-Cost Preference Judgments with Ties.” In: *The 3rd ACM International Conference on the Theory of Information Retrieval*. ICTIR ’17. ACM (cit. on pp. 2, 109).

Hui, Kai and Berberich, Klaus (2017c). “Transitivity, Time Consumption, and Quality of Preference Judgments in Crowdsourcing.” In: *The 39th European Conference on Information Retrieval*. ECIR ’17. Springer International Publishing, pp. 239–251 (cit. on pp. 2, 109).

Hui, Kai, Berberich, Klaus, and Mele, Ida (2017a). “Dealing with Incomplete Judgments in Cascade Measures.” In: *The 3rd ACM International Conference on the Theory of Information Retrieval*. ICTIR ’17. ACM (cit. on pp. 2, 109).

Hui, Kai, Yates, Andrew, Berberich, Klaus, and de Melo, Gerard (2018). “Co-PACRR: A Context-Aware Neural IR Model for Ad-hoc Retrieval.” In: *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. WSDM ’18. Los Angeles, California, USA: ACM (cit. on pp. 2, 109).

Hui, Kai, Yates, Andrew, Berberich, Klaus, and Melo, Gerard de (2017b). “PACRR: A Position-Aware Neural IR Model for Relevance Matching.” In: *Proceedings of the 2017 Conference on*

Empirical Methods in Natural Language Processing. EMNLP '17. Copenhagen, Denmark: Association for Computational Linguistics (cit. on pp. 2, 83, 88, 89, 97, 109).

Hui, Kai, Yates, Andrew, Berberich, Klaus, and Melo, Gerard de (2017c). "Position-Aware Representations for Relevance Matching in Neural Information Retrieval." In: *Proceedings of the 26th International Conference on World Wide Web Companion*. WWW '17. International World Wide Web Conferences Steering Committee, pp. 799–800 (cit. on pp. 2, 109).

Huston, Samuel and Croft, W. Bruce (2014). "A Comparison of Retrieval Models using Term Dependencies." In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. Ed. by Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang. CIKM '14. ACM, pp. 111–120 (cit. on p. 81).

Jardine, Nick and Rijsbergen, Cornelis Joost van (1971). "The use of hierarchic clustering in information retrieval." In: *Information storage and retrieval* 7.5, pp. 217–240 (cit. on p. 22).

Järvelin, Kalervo and Kekäläinen, Jaana (2002). "Cumulated gain-based evaluation of IR techniques." In: *ACM Transactions on Information Systems* 20 (4), pp. 422–446. ISSN: 1046-8188 (cit. on pp. 9, 12, 13, 92).

Jelinek, Frederick (1980). "Interpolated estimation of Markov source parameters from sparse data." In: *Proc. of the Workshop on Pattern Recognition in Practice*, pp. 381–397 (cit. on p. 8).

Joachims, Thorsten (1998). "Text categorization with support vector machines: Learning with many relevant features." In: *Machine learning: ECML-98*, pp. 137–142 (cit. on p. 17).

Kalchbrenner, Nal, Grefenstette, Edward, and Blunsom, Phil (2014). "A convolutional neural network for modelling sentences." In: *arXiv preprint arXiv:1404.2188* (cit. on p. 88).

Kazai, Gabriella, Yilmaz, Emine, Craswell, Nick, and Tahaghoghi, Seyed MM (2013). "User intent and assessor disagreement in web search evaluation." In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. CIKM '13. ACM, pp. 699–708 (cit. on pp. 2, 53, 54, 56, 57, 71, 75, 76).

Landauer, Thomas K, Foltz, Peter W, and Laham, Darrell (1998). "An introduction to latent semantic analysis." In: *Discourse processes* 25.2-3, pp. 259–284 (cit. on pp. 16, 23, 27).

Le, Quoc V and Mikolov, Tomas (2014). "Distributed representations of sentences and documents." In: *arXiv preprint arXiv:1405.4053* (cit. on pp. 16, 23, 27, 28).

MacKay, David JC and Peto, Linda C Bauman (1995). "A hierarchical Dirichlet language model." In: *Natural language engineering* 1.03, pp. 289–308 (cit. on p. 8).

Manning, Christopher D, Raghavan, Prabhakar, Schütze, Hinrich, et al. (2008). *Introduction to information retrieval*. Vol. 1. 1. Cambridge university press Cambridge (cit. on pp. 6, 9).

Metzler, Donald and Croft, W Bruce (2005). "A Markov random field model for term dependencies." In: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '05. ACM, pp. 472–479 (cit. on pp. 6, 81).

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff (2013). "Distributed representations of words and phrases and their compositionality." In: *Advances in neural information processing systems*, pp. 3111–3119 (cit. on pp. 16, 27).

Mitra, Bhaskar, Diaz, Fernando, and Craswell, Nick (2017). "Learning to Match using Local and Distributed Representations of Text for Web Search." In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '16. International World Wide Web Conferences Steering Committee, pp. 1291–1299 (cit. on pp. 81–83, 86, 87, 92).

Mitra, Bhaskar, Nalisnick, Eric, Craswell, Nick, and Caruana, Rich (2016). "A dual embedding space model for document ranking." In: *arXiv preprint arXiv:1602.01137* (cit. on p. 89).

Moffat, Alistair and Zobel, Justin (2008). "Rank-biased Precision for Measurement of Retrieval Effectiveness." In: *ACM Trans. Inf. Syst.* 27.1, 2:1–2:27. ISSN: 1046-8188 (cit. on pp. 13, 14, 40).

Nair, Vinod and Hinton, Geoffrey E (2010). "Rectified linear units improve restricted boltzmann machines." In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. ICML '10, pp. 807–814 (cit. on p. 18).

Nemhauser, G.L., Wolsey, L.A., and Fisher, M.L. (1978). "An analysis of approximations for maximizing submodular set functions—I." English. In: *Mathematical Programming* 14 (1), pp. 265–294. ISSN: 0025-5610 (cit. on p. 31).

Niu, Shuzi, Guo, Jiafeng, Lan, Yanyan, and Cheng, Xueqi (2012). "Top-k learning to rank: labeling, ranking and evaluation." In: *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '12. ACM, pp. 751–760 (cit. on pp. 54, 57).

Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., and Lioma, C. (2006). "Terrier: A High Performance and Scalable Information Retrieval Platform." In: *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*. Seattle, Washington, USA (cit. on p. 92).

Pang, Liang, Lan, Yanyan, Guo, Jiafeng, Xu, Jun, and Cheng, Xueqi (2016a). "A Study of MatchPyramid Models on Ad-hoc Retrieval." In: *CoRR* abs/1606.04648 (cit. on pp. 81–83, 92, 93).

Pang, Liang, Lan, Yanyan, Guo, Jiafeng, Xu, Jun, Wan, Shengxian, and Cheng, Xueqi (2016b). "Text Matching As Image Recognition." In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16, pp. 2793–2799 (cit. on p. 85).

Pavlu, V and Aslam, J (2007). *A practical sampling strategy for efficient retrieval evaluation*. Tech. rep. Technical report, Northeastern University (cit. on pp. 24, 30, 33, 34).

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12, pp. 2825–2830 (cit. on p. 34).

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D (2014). "Glove: Global Vectors for Word Representation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Vol. 14. EMNLP '14, pp. 1532–1543 (cit. on p. 16).

Phan, Xuan-Hieu and Nguyen, Cam-Tu (2007). “GibbsLDA++: AC/C++ implementation of latent Dirichlet allocation (LDA).” In: *Tech. rep.* (Cit. on p. 23).

Platt, John (1998). “Sequential minimal optimization: A fast algorithm for training support vector machines.” In: (cit. on p. 17).

Ponte, Jay M and Croft, W. Bruce (1998). “A language modeling approach to information retrieval.” In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR ’98. ACM, pp. 275–281 (cit. on p. 7).

Radinsky, Kira and Ailon, Nir (2011). “Ranking from Pairs and Triplets: Information Quality, Evaluation Methods and Query Complexity.” In: *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*. WSDM ’11. Hong Kong, China: ACM, pp. 105–114. ISBN: 978-1-4503-0493-1 (cit. on pp. 53, 56, 59).

Řehůřek, Radim and Sojka, Petr (2010). “Software Framework for Topic Modelling with Large Corpora.” English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, pp. 45–50 (cit. on p. 23).

Rijsbergen, C. J. Van (1979). *Information Retrieval*. 2nd. Newton, MA, USA: Butterworth-Heinemann. ISBN: 0408709294 (cit. on pp. 22, 25).

Robertson, Stephen E (1977). “The probability ranking principle in IR.” In: *Journal of documentation* 33.4, pp. 294–304 (cit. on p. 5).

Robertson, Stephen E (2004). “Understanding inverse document frequency: on theoretical arguments for IDF.” In: *Journal of documentation* 60.5, pp. 503–520 (cit. on p. 6).

Robertson, Stephen E and Walker, Steve (1994). “Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval.” In: *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR ’94. Springer-Verlag New York, Inc., pp. 232–241 (cit. on p. 6).

Rorvig, Mark E (1990). “The simple scalability of documents.” In: *Journal of the American Society for Information Science* 41.8, p. 590 (cit. on pp. 53, 55, 56, 76, 79).

Sakai, Tetsuya (2007). “Alternatives to bpref.” In: *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR ’07. ACM, pp. 71–78 (cit. on pp. 22, 24, 25).

Sakai, Tetsuya (2013). “The Unreusability of Diversified Search Test Collections.” In: *EVIA@ NTCIR* (cit. on p. 25).

Sakai, Tetsuya, Dou, Zhicheng, Song, Ruihua, and Kando, Noriko (2012). “The reusability of a diversified search test collection.” In: *Asia Information Retrieval Symposium*. AIR ’12. Springer, pp. 26–38 (cit. on pp. 22, 25, 41, 42).

Severyn, Aliaksei and Moschitti, Alessandro (2015). “Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks.” In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’15 (cit. on p. 85).

Shen, Yelong, He, Xiaodong, Gao, Jianfeng, Deng, Li, and Mesnil, Grégoire (2014). “Learning Semantic Representations Using Convolutional Neural Networks for Web Search.” In: *Pro-*

ceedings of the 23rd International Conference on World Wide Web. WWW '14 Companion (cit. on pp. 81, 85).

Song, Ruihua, Guo, Qingwei, Zhang, Ruochi, Xin, Guomao, Wen, Ji-Rong, Yu, Yong, and Hon, Hsiao-Wuen (2011). "Select-the-Best-Ones: A new way to judge relative relevance." In: *Information processing & management* 47.1, pp. 37–52 (cit. on pp. 53–58, 61, 65, 79).

Tao, Tao and Zhai, ChengXiang (2007). "An exploration of proximity measures in information retrieval." In: *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '07. ACM, pp. 295–302 (cit. on pp. 6, 81, 83).

Tong, Simon and Koller, Daphne (2001). "Support vector machine active learning with applications to text classification." In: *Journal of machine learning research* 2.Nov, pp. 45–66 (cit. on p. 61).

Tong, Simon and Koller, Daphne (2002). "Support vector machine active learning with applications to text classification." In: *The Journal of Machine Learning Research* 2, pp. 45–66 (cit. on p. 64).

Voorhees, Ellen M. (1985). "The Cluster Hypothesis Revisited." In: *Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '85. Montreal, Quebec, Canada: ACM, pp. 188–196. ISBN: 0-89791-159-8 (cit. on p. 28).

Voorhees, Ellen M. (2001). "Evaluation by Highly Relevant Documents." In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '01. New Orleans, Louisiana, USA: ACM, pp. 74–82. ISBN: 1-58113-331-6 (cit. on p. 41).

Vu, Huyen-Trang and Gallinari, Patrick (2006). "A Machine Learning Based Approach to Evaluating Retrieval Systems." In: *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. HLT-NAACL '06. New York, New York: Association for Computational Linguistics, pp. 399–406 (cit. on pp. 23, 24).

Wang, Jiannan, Li, Guoliang, Kraska, Tim, Franklin, Michael J, and Feng, Jianhua (2013). "Leveraging transitive relations for crowdsourced joins." In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. SIGMOD '13. ACM, pp. 229–240 (cit. on p. 62).

Xu, Jinxi and Croft, W. Bruce (1996). "Query Expansion Using Local and Global Document Analysis." In: *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'96, August 18-22, 1996, Zurich, Switzerland (Special Issue of the SIGIR Forum)*. Ed. by Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson. SIGIR'96. ACM, pp. 4–11 (cit. on p. 82).

Yilmaz, Emine and Aslam, Javed A. (2006). "Estimating Average Precision with Incomplete and Imperfect Judgments." In: *Proceedings of the 15th ACM International Conference on*

Information and Knowledge Management. CIKM ’06. Arlington, Virginia, USA: ACM, pp. 102–111. ISBN: 1-59593-433-2 (cit. on pp. 25, 34).

Yu, Kai, Bi, Jinbo, and Tresp, Volker (2006). “Active learning via transductive experimental design.” In: *Proceedings of the 23rd international conference on Machine learning*. ICML ’06. ACM, pp. 1081–1088 (cit. on pp. 22, 24).

Zamani, Hamed and Croft, W. Bruce (2017). “Relevance-based Word Embedding.” In: *arXiv preprint arXiv:1705.03556* (cit. on pp. 83, 89).

Zhai, ChengXiang (2008a). “Statistical Language Models for Information Retrieval A Critical Review.” In: *Found. Trends Inf. Retr.* 2 (3), pp. 137–213. ISSN: 1554-0669 (cit. on p. 7).

Zhai, ChengXiang (2008b). “Statistical Language Models for Information Retrieval A Critical Review.” In: *Found. Trends Inf. Retr.* 2 (3), pp. 137–213. ISSN: 1554-0669 (cit. on p. 38).

Zhai, ChengXiang and Lafferty, John (2002). “Two-stage language models for information retrieval.” In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR ’02. ACM, pp. 49–56 (cit. on p. 8).

Zhai, Chengxiang and Lafferty, John (2001). “A study of smoothing methods for language models applied to ad hoc information retrieval.” In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR ’01. ACM, pp. 334–342 (cit. on pp. 8, 85).

Zhu, Dongqing and Carterette, Ben (2010). “An analysis of assessor behavior in crowdsourced preference judgments.” In: *SIGIR 2010 workshop on crowdsourcing for search evaluation*, pp. 17–20 (cit. on pp. 53, 56).

List of Figures

List of Tables