Effective Collection Construction for Information Retrieval Evaluation and Optimization

Dan Li

Effective Collection Construction for Information Retrieval Evaluation and Optimization

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op vrijdag 20 november 2020, te 10:00 uur

door

Dan Li

geboren te Nei Mongol

Promotiecommissie

Promotor:		
	Prof. dr. E. Kanoulas	Universiteit van Amsterdam
Co-promotor:		
	Prof. dr. M. de Rijke	Universiteit van Amsterdam
Overige leden:		
	prof. dr. Y. Liu	Tsinghua University
	dr. I. Markov	Universiteit van Amsterdam
	prof. dr. J. Mothe	Institut de Recherche en Informatique de Toulouse
	prof. dr. C.G.M. Snoek	Universiteit van Amsterdam
	prof. dr. M. Worring	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The research was supported by the China Scholarship Council.

Cover by Feng Li
Printed by Offpage, Amsterdam

ISBN: 978-94-93197-29-9

Contents

1 Introduction

Machine learning is widely applied in research fields such as information retrieval (IR), natural language processing, and computer vision. Constructing high-quality datasets plays a crucial role in developing machine learning models. At the same time, it has become one of the biggest bottlenecks, not only for the research community, but also for industry or governmental organizations [130]. Why has dataset construction always been a critical issue? One obvious reason is that new tasks and new applications of machine learning emerge. The research field is moving fast, the number of publications in 2018 has increased about 30 times since 2009 [47]. Meanwhile, machine learning models, especially deep learning models, are becoming bigger and more complex, usually requiring large amounts of labeled data to be trained. The dependence of machine learning models on large labeled training data has led to various techniques for dealing with a lack of labeled training data, such as active learning, weak or distant supervised learning, semi-supervised learning, self-supervised learning, transfer learning, and multi-task learning [130].

It is instructive to look at the history of IR techniques through the lens of dataset construction. IR as a discipline dates back to the 1940s during the first burgeoning of publications [170]. The Cranfield collection was a pioneering dataset in allowing precise quantitative measures of indexing and retrieval effectiveness. It grew out of the Cranfield project [32–34, 79], which marks a transition from IR as a traditional theoretical field of study to rigorous empirical experiments in IR. Two crucial questions for the Cranfield project were, first, how to generate the retrieval requests (*information needs*), and second, how to determine whether a request has been addressed successfully by the output of a search, or how to evaluate *retrieval effectiveness*. The project collected 18,000 papers on aeronautical science as the *document collection*, from which a set of papers were selected and the corresponding authors were asked to write down the original research questions. These research questions, 225 in total, served as the retrieval requests. In order to save on assessment costs, 1,400 papers were selected for exhaustive *relevance assessment*. Then, search was conducted using different indexing schemes. The effectiveness of the search results was evaluated by *precision* and *recall*.

The Cranfield paradigm of controlled laboratory experiments set the foundations of modern evaluation methodology of IR. Since then, test collections for evaluation in IR typically contain three parts: (1) a *document collection*, against which the search is conducted; (2) a set of *information needs*, expressed as *topics* or *queries*, defining the

search goal; and (3) a set of *relevance assessments*, typically binary labels of relevance for query-document pairs. Relevance is assessed relative to an information need, not a query.

Later projects, like SMART [138], followed the evaluation framework of Cranfield. One important contribution of SMART was that it digitized the Cranfield and other datasets and demonstrated the possibilities of fully automated evaluation [138]. Most follow-up work has demonstrated a similar conclusion as the Cranfield experiments namely that simple term indexing and a simple statistical model work well. All early work uses scientific papers as the collection, and the sizes of the collections are rather small, around 1,000 to 2,000 abstracts. Constructing such a test collection can be as expensive as $US 800,000 in today's values, mostly spent on document digitization [170].

Since 1992, the text retrieval conference (TREC) has organized many tracks over a range of different test collections, aiming to provide a large-scale, collaborative, and comparative experimental platform. The most representative ones are the first 8 ad-hoc tracks. The test collections used in those tracks contain 1.89 million newswire documents and relevance assessments for 450 topics [162]. The large-scale relevance assessments were carried out by retired intelligence agents.

The important contribution of TREC does not only concern the number and size of the collections used, but also the experimental methodology that was developed. For example, *pooling* was introduced as a mean to effectively assess relevance. Due to the large scale of document collections, it was infeasible to do exhaustive relevance assessments. Rather, relevance was assessed over a subset of the collection that is formed from the top k documents returned by a number of different retrieval systems that participated in a retrieval experiment. Pooling was shown to achieve reasonable coverage of the relevance set and allowed the collection to be re-used [184]. The pooling method has since become the de facto method for constructing IR test collections. Later, the further increase of the size of test collections and the further development of retrieval techniques revealed some drawbacks of pooling. For example, the underestimation of recall and the pooling bias generated when re-using pooled collections to evaluate novel systems that retrieve relevant but unassessed documents are well-known problems [23, 107, 169, 185]. The literature suggests a number of approaches to cope with missing assessments by defining IR measures that are robust to missing assessments [25] or introducing document selection approaches that carefully choose which documents to be assessed. An overview can be found in [80, 139].

The growth of TREC also witnessed the transition from stand-alone search systems to web search services. Since the appearance of the first website of CERN in 1991, the volume of content on the web has been increasing with a high rate, making the ad-hoc collections quickly fall out of date. Several web tracks have been proposed in order to address issues that web search was facing such as the large-scale and heterogeneous content indexing, document link analysis, search log analysis, user behavior modelling, and spam detection. Representative collections include, but are not limited to the GOV2 collection, which contains 25 million web pages, the ClueWeb09 and ClueWeb12 collection, which contains 1 billion web pages in ten languages.

Search has now become ubiquitous. It is present in many environments, such as desktop, enterprise, and in many domains, such as medical, legal, product, expert, so-

cial media. Each of these environments or domains requires its own test collection. Relevance assessment previously conducted by experts became rather expensive and slow to obtain. Crowdsourcing arose as a popular cost-effective solution for the construction of test collections around 2010 [22]. Platforms like Amazon Mechanical Turk provide a service of distributing tasks (called HITs) to human workers, who are compensated for finishing the tasks. Crowdsourcing was investigated by the IR community, in the TREC crowdsourcing track [24, 146] and the TREC core track [1]. It made relevance assessment easy and fast to obtain, but at the same time it caused new issues such as disagreement between crowd workers and the appearance of spam workers. The concept of relevance is often treated as an objective attribute of a document with regard to an information need, which is evidenced by the popularity of user-independent test collections in IR community. However, a crowd worker has his or her own understanding of the information need and document content, and different workers may report idiosyncratic and variable subjective assessments of relevance [2]. It is unlikely to achieve perfect agreement between different crowd workers on the same query-document pair. The literature on crowd relevance assessments in IR copes with many important issues like the relationship between the quality of crowd relevance assessments and the evaluation in IR [3, 4, 114], human factors affecting the quality of crowd relevance assessments [88], the rationale and strategy of crowd workers [62, 118], and aggregating noisy crowd relevance assessments for high-quality test collection [22].

With the emergence of new tasks in IR, the need to construct new test collections is a continuous demand. For example, the number of topics was set to 50 in the original ad-hoc test collections of TREC, but increased to thousands in the TREC million-query test collection in web search scenario. Besides, topics have been evolving form traditional single-turn ad hoc search to multi-turn conversational search [44]. Further, the domain of new collections has expanded to include to medicine, law, social media and genomics. One of the advantages of having a standard test collection is that there is a fixed setting in which we can vary retrieval systems and system parameters to carry out comparative experiments. Such formal testing is less expensive compared with manual evaluation and allows for a clearer diagnosis of the effect of changing system parameters than conducting user studies on retrieval effectiveness. Furthermore, we can optimize the effectiveness of retrieval system by machine learning methods rather than manually tuning parameters, if we have a formal measure that we have confidence in. Automatic techniques for optimizing model parameters have attracted the attention of the research community in recent years [58, 64–66, 111, 136, 143].

The goal of the work in the thesis is to provide a fundamental way of constructing and utilizing test collections in information retrieval in an effective, efficient and reliable manner. To that end, we focus on four aspects around test collections for information retrieval:

1. Document selection with the objective of identifying relevant documents; this addresses an important issue of the *pooling* method.
2. Stopping document selection when both the gain of identifying relevant documents and the cost of assessing documents are considered as the optimization goals; this is an important but understudied problem in the construction of test collections.
3. Denoising relevance assessments by aggregating from multiple crowd annotation sources to obtain high-quality relevance assessments; this helps to boost the quality

of relevance assessments acquired in a crowdsourcing manner.
4. Optimizing the configuration of retrieval systems on the basis of existing test collections and effectiveness measures.

In the next section we outline the research in this thesis and the questions that are answered within it.

1.1 Research Outline and Questions

This thesis contains two research themes: constructing test collections for evaluating retrieval systems (Chapter 2, 3 and 4) and optimizing the configuration of retrieval systems on the basis of test collections (Chapter 5). Below, we elaborate the main research question of every chapter.

1.1.1 Evaluation

Evaluation is crucial in IR. The development of models, tools and methods has significantly benefited from the availability of reusable test collections formed through a standardized and thoroughly tested methodology, known as the Cranfield paradigm [31], as we pointed out earlier. When the document collection is large, identifying all relevant documents is difficult due to the immense human labor required. In order to avoid assessing the entire document collection, depth-k pooling [151] is often used. Pooling aims at being fair to all runs and aims for a diverse set of submitted runs that can provide a good coverage of all relevant documents. Nevertheless, the underestimation of recall and the pooling bias generated when re-using pooled collections to evaluate novel systems that retrieve relevant but unassessed documents are well-known problems [23, 107, 169, 185]. One direction towards solving these issues is by introducing a document selection methodology that carefully chooses which documents to be assessed. Methods proposed under this approach belong to two categories: (1) sampling-based methods [10, 126, 142, 172, 174], and (2) active selection methods [11, 43, 106, 109]. We hope to reduce the variance of the sampling approaches and avoid the bias of the active selection methods towards good systems. Therefore the first research question is:

RQ1 How can we effectively select documents in order to construct a test collection that allows for unbiased and low-variance effectiveness measures?

To answer **RQ1**, we devise an *active sampling* method. We follow a sampling-based approach for efficient large-scale evaluation. Different from past sampling-based approaches, we account for the fact that some systems are of higher quality than others, and we design our sampling distribution to over-sample documents from these systems. At the same time, given that our approach is a sampling-based approach, the estimated evaluation measures are, by construction, unbiased on average, and assessments can be used to evaluate new, novel systems without introducing any systematic error.

One of the goals of the Cranfield paradigm test collection is to have relevance assessments that are complete. Moreover, this demand is an important desideration in IR applications like electronic discovery, systematic review, investigation, and research [41, 42, 60, 81, 83, 85, 94, 135], which are known to be *high recall* or *total recall* tasks.

To identify as many relevant documents as possible, active-learning approaches have been employed and shown their effectiveness [35, 36, 38, 39]. Note that dealing with a high recall task is not the same as constructing test collections. The main goal of constructing a test collection is to accurately evaluate the ranking performance of retrieval systems. A complete relevance assessment of all documents in the collection is one way to achieve the goal, but sampling based methods also allow accurate evaluation by sampling a subset of the documents and estimating unbiased effectiveness measures based on the sample. In this part of the thesis we work towards the first direction and purely focus on achieving high recall rather than accurate evaluation. Obviously, there is a trade-off between recall and assessment cost in all these active-learning approaches. A natural question is "when to stop selecting documents for assessment." This is an under-studied issue and only a few publications have tackled the problem. Some work [35, 50, 110] proposes heuristic rules to determine an ad-hoc stopping point, which is lacking transparency because there is no indication on how many relevant documents are still not found. Other work [37, 165] requires extra assessment cost to sample documents to estimate the total number of relevant document in the collection. None of the methods can achieve high recall with low assessment cost and provide transparency at the same time. Therefore, the second research question is:

RQ2 How can we select documents for assessing relevance and stopping the selection, both in an effective manner to maximize recall and minimize assessment cost and in a transparent manner so that we know the number of residual relevant documents?

To answer **RQ2**, we handle the problem of deciding the stopping point under the continuous active learning framework by jointly training a ranking model to rank documents, and estimating the total number of relevant documents in the collection using a "greedy" sampling method. We prove the unbiasedness of the proposed estimators under a with-replacement sampling design. Our experimental results demonstrate that the proposed approach, similar to the continuous active learning approach [35], effectively retrieves relevant documents but it also provides a transparent, accurate, and effective stopping point.

The next stage of constructing a test collection is assessing relevance. Crowdsourcing has arisen as a natural cost-effective solution. Despite its cost-effectiveness, crowdsourcing introduces a major challenge – controlling the quality of the relevance labels through careful aggregation of crowd labels [92]. Majority voting has been the most prominent aggregation method, with early experiments showing that labels derived by a majority vote of multiple untrained crowd annotators can reach a quality comparable to that of a trained expert assessor [4]. At the same time, recent work has shown that the quality of annotations is affected by a number of factors that are not considered by majority voting, two of the most important being the inherent *difficulty* of the crowdsourcing task and the annotator's *competence* to complete the task [53]. The third research question is:

RQ3 How can we effectively aggregate crowdsourcing labels in order to acquire high-quality labels in test collections?

Most current work that aims to infer the true relevance of a query-document pair from

such noisy annotations, however, assumes that the tasks are independent from each other, and that the annotators are independent from each other as well. To answer **RQ3**, we relax this assumption and instead assume a Gaussian process prior on the latent true labels to model the correlation between tasks. The proposed multi-annotator Gaussian process model is able to model the latent true labels, the task bias and variance (difficulty), and the annotator bias and variance (competence).

1.1.2 Optimization

The effectiveness of IR systems heavily depends on a large number of configurations that need to be tuned [28, 57]. Configurations range from the choice of different system components, e.g., stopword lists, stemming methods, retrieval models, to model parameters. Zhai et al. [177] demonstrated this sensitivity for the smoothing parameters in language models, and Trotman et al. [159] for parameters in the BM25 model. Automatic techniques for optimizing model configurations have attracted the attention of the research community in recent years [58, 64–66, 111, 136, 143]. However, configurations are usually optimized in isolation, while their mutual dependencies are, to a great extend, unexplored [7, 90]. The fourth research question is:

RQ4 How can we optimize the configuration of retrieval systems with regard to effectiveness measures on the basis of existing test collections?

To answer **RQ4**, we use Bayesian optimization (BO) to automatically optimize retrieval system configurations. Information retrieval effectiveness, however, as a function of the configuration space exhibits high irregularity. Furthermore, the values of some configurations can be continuous, or categorical [52], with the latter contributing most of the irregularity of the objective function. To tackle this we model the effect of a δ-step in the configuration space to the effectiveness of the retrieval system, by suggesting to use different similarity functions (covariance functions) for continuous and categorical values, and examine their ability to effectively and efficiently guide the search in the configuration space.

1.2 Main Contributions

In this section, we list theoretical, algorithmic and empirical contributions of the thesis. For each contribution, we list the chapter from which it originates.

1.2.1 Theoretical Contributions

1. An active sampling method for selecting documents for the construction of a test collection and the evaluation of retrieval systems. We name it the active sampling (AS) method (Chapter 2).

2. A continuous active learning and sampling framework for selecting documents and determining the stopping point of document selection for constructing test collections that balance the cost of assessing document relevance and the gain of identifying relevant documents. We name it the automatic stopping (AUTOSTOP) framework. A proof of the unbiasedness of the proposed estimators of the total number of documents under our sampling design (Chapter 3).

3. A probabilistic model based on Gaussian processes to aggregate crowd relevance assessments for test collection construction. We name it the multi-annotator Gaussian process (MAGP) model. A variational expectation maximization algorithm is applied to learn the model parameters (Chapter 4).

4. A Bayesian-optimization-based model for optimizing the configuration of retrieval systems. We name it the Bayesian optimization for information retrieval (BO4IR) method (Chapter 5).

1.2.2 Empirical Contributions

5. (1) An empirical verification of the effectiveness of our AS method compared with sampling-based and active-selection methods regarding bias and variance in the calculated effectiveness measures. (2) An empirical comparison and analysis of our estimators with sampling-based and active-selection methods regarding approximating the actual evaluation measures. (3) An empirical comparison and analysis of our AS method with sampling-based and active-selection methods regarding the reusability of the constructed test collection. (4) An implementation of baseline methods including the move-to-front method [43], the multi-armed bandit method [109], and the stratified sampling method [126] (Chapter 2).

6. (1) An empirical verification of the effectiveness of our AUTOSTOP framework on various datasets regarding maximizing recall and minimizing assessment cost with knowing the number of residual relevant documents. (2) An empirically verification of the unbiasedness of the proposed estimators of total number of documents. (3) An empirical analysis of the estimation module and the stopping module of our framework. (4) An implementation of baseline methods including the knee method [37], the target method [37], the scalable continuous active learning method [39], and the score distribution method [71] (Chapter 3).

7. (1) An empirical verification of the effectiveness of our MAGP model regarding inferring true relevance labels of query-document pairs, on both synthetic data and real data, compared with extensive baselines. (2) An empirical analysis of the impact of the annotation quality of crowdsourcing datasets on the performance of our MAGP model and an empirical analysis of the impact of the query-document representation module of our MAGP model on its performance. (Chapter 4).

8. (1) An empirical demonstration of the effectiveness of our BO4IR method in building IR systems and our analysis of the reason behind its effectiveness via observing optimization behavior in the search space. (2) An empirical analysis of the components of our BO4IR method including initialization, covariance function, acquisition function and selection strategy (Chapter 5).

1.2.3 Dataset and Software Contributions

As part of the research we carried out for the thesis, we contributed three new test collections and new code of our proposed models to the community.

9. A dataset in the 2017 CLEF Technology-Assisted Reviews in Empirical Medicine track, which consists of topics from 50 Diagnostic Test Accuracy systematic re-

views, a subset of PubMed document collection, together with the corresponding relevance labels are released (Chapter 3).

10. A dataset in the 2018 CLEF Technology-Assisted Reviews in Empirical Medicine track, which consists of topics from 30 new systematic reviews of Diagnostic Test Accuracy, a subset of PubMed document collection, together with the corresponding relevance labels are released (Chapter 3).
11. A dataset in the 2019 CLEF Technology-Assisted Reviews in Empirical Medicine track, which consists of topics from 8 new reviews of diagnostic test accuracy, 20 new reviews of intervention, 1 new review of prognosis, and 2 new systematic reviews of quality, a subset of PubMed document collection, together with the corresponding relevance labels are released (Chapter 3).
12. The code of the AS method is released under an open source license (Chapter 2).
13. The code of the baseline methods, (the knee method [37], the target method [37], the scalable continuous active learning method [39], the score distribution method [71]), and our AUTOSTOP method is released under an open source license (Chapter 3).
14. The code of our MAGP model is released under an open source license (Chapter 4).
15. The code of our BO4IR method is released under an open source license (Chapter 5).

1.3 Thesis Overview

This thesis is organized into two parts: (1) constructing test collections for evaluating retrieval systems, and (2) optimizing the configuration of retrieval systems on the basis of the test collection.

Figure 1.1 provides the high-level overview of the thesis. It illustrates a high level structure of several classical modules related to IR collections. Topics, document collections and relevance labels are three key concepts in the Cranfield test collection. These concepts are related to important tasks such as constructing topics, selecting documents, labelling document relevance, evaluating retrieval systems, building retrieval systems, etc. In this thesis, we tackle the following tasks: document selection, relevance labelling and optimizing retrieval systems.

The first part of the thesis consists of three chapters. In Chapter 2, we address the issue of selecting documents in order to construct test collections that reduce assessment cost and allow unbiased and low-variance measures. In Chapter 3, we address the issue of stopping the selection of documents in order to construct test collections that balance the cost of assessing document relevance and the gain of identifying relevant documents. In Chapter 4, we address the issue of aggregating crowdsourcing labels in order to acquire high-quality labels in test collections.

The second part of the thesis consists of one chapter. In Chapter 5, we address the issue of optimizing the configuration of retrieval systems on the basis of existing test collections.

Finally, in Chapter 6, we conclude the thesis and discuss limitations and future directions.

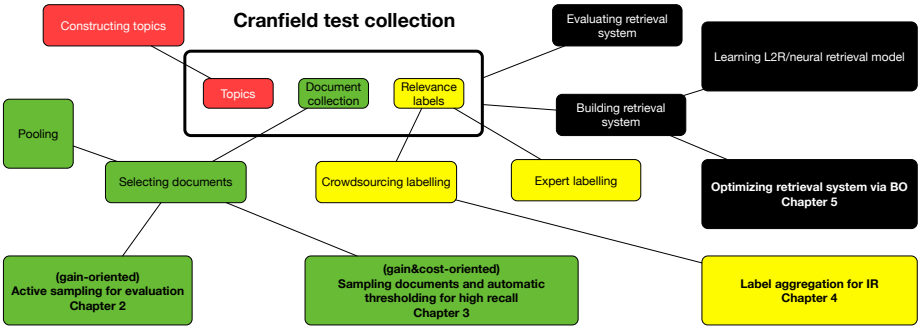


Figure 1.1: A high level structure of several classic modules related to IR test collections. In this thesis, we tackle the four tasks marked by bold fonts.

1.4 Origins

In this section, we list the publications which each chapter is based on and explain the role of each author.

- **Chapter 2** is based on the following paper:
  - Dan Li and Evangelos Kanoulas. 2017. Active sampling for large-scale information retrieval evaluation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (CIKM '17). Association for Computing Machinery, Singapore, Singapore, 49–58.

  DL designed the model, EK helped with the design of the model; DL ran the experiments and did most of the writing; EK helped with the writing.

- **Chapter 3** is based on the following papers:
  - Dan Li and Evangelos Kanoulas. 2020. When to stop reviewing in technology-assisted reviews. *The ACM Transactions on Information Systems*. Accepted, to appear.

  DL designed the model, ran the experiments and did most of the writing; EK helped with the writing.

  - Evangelos Kanoulas, Dan Li, Leif Azzopardi and René Spijker. 2017. CLEF 2017 technologically assisted reviews in empirical medicine overview. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017* (CEUR Workshop Proceedings). Volume 1866. CEUR-WS.org.

  EK, RS and LA proposed the tracks, DL constructed the test construction, EK, RS, LA and DL contributed to the writing.

  - Evangelos Kanoulas, Dan Li, Leif Azzopardi and René Spijker. 2018. CLEF 2018 technologically assisted reviews in empirical medicine overview. In *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018* (CEUR Workshop Proceedings). Volume 2125. CEUR-WS.org.

  EK, RS and LA proposed the tracks, DL constructed the test construction, EK, RS,

LA and DL contributed to the writing.

– Evangelos Kanoulas, Dan Li, Leif Azzopardi and René Spijker. 2019. CLEF 2019 technology assisted reviews in empirical medicine overview. In *Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum, Lugano, Switzerland, September 9-12, 2019* (CEUR Workshop Proceedings). Volume 2380. CEUR-WS.org.

EK, RS and LA proposed the tracks, DL constructed the test construction, EK, RS, LA and DL contributed to the writing.

- **Chapter 4** is based on the following papers:
  – Dan Li and Evangelos Kanoulas. 2020. A multi-annotator gaussian process model for relevance inference from noisy annotations. *The Web Conference 2021*. Under submission.

  DL designed the model, ran the experiments and did most of the writing; EK contributed to the writing.

- **Chapter 5** is based on the following paper:
  – Dan Li and Evangelos Kanoulas. 2018. Bayesian optimization for optimizing retrieval systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (WSDM '18). Association for Computing Machinery, Marina Del Rey, CA, USA, 360–368.

  DL and EK designed the model, DL ran the experiments and did most of the writing; EK helped with the writing.

The thesis also indirectly builds on the following papers.
Full/short/journal papers:

- Dan Li, Panagiotis Zafeiriadis and Evangelos Kanoulas. 2020. APS: an active PubMed search system for technology assisted review. In *The 43st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2020, Xi'an, China, July 25-30, 2020*. ACM.
- Jie Zou, Dan Li and Evangelos Kanoulas. 2018. Technology assisted reviews: finding the last few relevant documents by asking yes/no questions to reviewers. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*. ACM, 949–952.
- Oana Inel, Giannis Haralabopoulos, Dan Li, Christophe Van Gysel, Zoltán Szlávik, Elena Simperl, Evangelos Kanoulas and Lora Aroyo. 2018. Studying topical relevance with evidence-based crowdsourcing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. ACM, 1253–1262.
- Nikos Voskarides, Dan Li, Pengjie Ren, Evangelos Kanoulas and Maarten de Rijke. 2020. Query resolution for conversational search with limited supervision. In *The 43st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2020, Xi'an, China, July 25-30, 2020*. ACM.
- Nikos Voskarides, Dan Li, Andreas Panteli and Pengjie Ren. 2019. ILPS at TREC

2019 conversational assistant track. In *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019* (NIST Special Publication). Volume 1250. National Institute of Standards and Technology (NIST).
- Yukun Zheng, Dan Li, Zhen Fan, Yiqun Liu, Min Zhang and Shaoping Ma. 2018. T-reader: a multi-task deep reading comprehension model with self-attention mechanism. *Journal of Chinese Information Processing*, 11, 128.

Worknote papers:
- Liadh Kelly, Hanna Suominen, Lorraine Goeuriot, Mariana L. Neves, Evangelos Kanoulas, Dan Li, Leif Azzopardi, René Spijker, Guido Zuccon, Harrisen Scells and João R. M. Palotti. 2019. Overview of the CLEF ehealth evaluation lab 2019. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 10th International Conference of the CLEF Association, CLEF 2019, Lugano, Switzerland, September 9-12, 2019, Proceedings* (Lecture Notes in Computer Science). Volume 11696. Springer, 322–339.
- Dan Li and Evangelos Kanoulas. 2019. Automatic thresholding by sampling documents and estimating recall. In *Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum, Lugano, Switzerland, September 9-12, 2019* (CEUR Workshop Proceedings). Volume 2380. CEUR-WS.org.
- Hanna Suominen, Liadh Kelly, Lorraine Goeuriot, Aurélie Névéol, Lionel Ramadier, Aude Robert, Evangelos Kanoulas, René Spijker, Leif Azzopardi, Dan Li, Jimmy, João R. M. Palotti and Guido Zuccon. 2018. Overview of the CLEF ehealth evaluation lab 2018. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 9th International Conference of the CLEF Association, CLEF 2018, Avignon, France, September 10-14, 2018, Proceedings* (Lecture Notes in Computer Science). Volume 11018. Springer, 286–301.
- Christophe Van Gysel, Dan Li and Evangelos Kanoulas. 2017. ILPS at TREC 2017 common core track. In *Proceedings of The Twenty-Sixth Text REtrieval Conference, TREC 2017, Gaithersburg, Maryland, USA, November 15-17, 2017* (NIST Special Publication). Volume 500-324. National Institute of Standards and Technology (NIST).
- James Allan, Donna Harman, Evangelos Kanoulas, Dan Li, Christophe Van Gysel and Ellen M. Voorhees. 2017. TREC 2017 common core track overview. In *Proceedings of The Twenty-Sixth Text REtrieval Conference, TREC 2017, Gaithersburg, Maryland, USA, November 15-17, 2017* (NIST Special Publication). Volume 500-324. National Institute of Standards and Technology (NIST).

Part I

Evaluation

2

Documents Selection through Active Sampling

In Chapter 1, we have introduced the structure of the research in this thesis. In this chapter, we devise an active sampling method to solve the problem of large-scale retrieval evaluation. We answer the following question asked in Chapter 1:

RQ1 How can we effectively select documents in order to construct a test collection that allows for unbiased and low-variance effectiveness measures?

2.1 Introduction

Evaluation is crucial in IR as we explained in Chapter 1. The development of IR models, tools and methods has significantly benefited from the availability of reusable test collections formed through a standardized and thoroughly tested methodology, known as the Cranfield paradigm [31]. Under the Cranfield paradigm the evaluation of retrieval systems typically involves assembling a document collection, creating a set of information needs (topics), and identifying a set of documents relevant to the topics.

One of the simplifying assumptions made by the Cranfield paradigm is that the relevance assessments are complete, i.e., for each topic all relevant documents in the collection have been identified. When the document collection is large, identifying all relevant documents is difficult due to the immense human labor required. In order to avoid judging the entire document collection depth-k pooling [151] is being used: a set of retrieval systems (also called *runs*) ranks the document collection against each topic, and only the union of the top-k retrieved documents is being assessed by human assessors. Documents outside the depth-k pool are considered non-relevant. Pooling aims at being fair to all runs and aims for a diverse set of submitted runs that can provide a good coverage of all relevant documents. Nevertheless, the underestimation of recall [185] and the pooling bias generated when re-using these pooled collections to evaluate novel systems that retrieve relevant but unjudged documents [23, 107, 169, 185] are well-known problems.

The literature suggests a number of approaches to cope with missing assessments (an overview can be found in [80, 139]): (1) Defining IR measures that are robust to missing assessments, like bpref [25]. The developed measures, however, may not precisely capture the notion of retrieval effectiveness one requires, while some have been

shown to remain biased [172]. (2) Running a meta-experiment where runs are "left out" from contributing to the pool and measuring the bias experienced by these left-out runs compared to the original pool, which is then used to correct measurements over new retrieval systems [104, 105, 107, 169]. (3) Leaving the design of the evaluation measure unrestricted, but instead introducing a document selection methodology that carefully chooses which documents to be judged. Methods proposed under this approach belong to two categories: (1) sampling-based methods [10, 126, 142, 172, 174], and (2) active selection methods [11, 43, 106, 109].

Sampling-based methods devise a sampling strategy that randomly selects a subset of documents to be assessed; evaluation measures are then inferred on the basis of the obtained sample. Different methods employ different sampling distributions. Aslam et al. [10] and Yilmaz et al. [172] use a uniform distribution over the ranked document collection, while Pavlu et al. [126] and Yilmaz et al. [174] recognize that relevant documents typically reside at the top of the ranked lists returned by participating runs and use stratified sampling to draw a larger sample from the top ranks. Schnabel et al. [142] also use a weighted-importance sampling method on documents with the sampling distribution optimized for a comparative evaluation between runs. In all the aforementioned work, an experiment that dictates the probability distribution under which documents are being sampled is being designed in such a way that evaluation measures can be defined as the expected outcome of this experiment. Evaluation measures can then be estimated by the judged documents sampled. In all cases the sampling distribution is being defined at the beginning of the sampling process and remains fixed throughout the experiment. Sampling-based methods have the following desirable properties: (1) on average, estimates have no systematic error; (2) past data can be re-used by new, novel runs without introducing bias; and (3) sampling distributions can be designed to optimize the number of assessments needed to confidently and accurately estimate a measure.

In contrast, active-selection methods recognize that systems contributing documents to the pool vary in quality. Based on this observation they bias the selection of documents towards those retrieved by good retrieval systems. The selection process is deterministic and depends on how accurately the methods can estimate the quality of each retrieval system. Judging is performed in multiple rounds: in each round the best system is identified, and the next unjudged document in the ranked list of this system is selected to be judged. The quality of systems is calculated at the end of each round, as soon as a new judgment becomes available. Active-selection methods include move-to-front [43], fixed-budget pooling [106], and multi-armed bandits [109]. Losada et al. [109] consider the problem as an exploration-exploitation dilemma, balancing between selecting documents from the best-quality run, and exploring the possibility that the quality of some runs might be underestimated in different rounds of the experiment. The advantage of active-selection methods compared to sampling-based methods is that they are designed to identify as many relevant document as possible, by selecting documents with the highest relevance probability. The disadvantage is that the judging process is not fair to all runs, with the selection of documents being biased towards good-performing runs.

In this chapter, we follow a sampling-based approach for an efficient large-scale evaluation. Different from past sampling-based approaches we account for the fact

that some systems are of higher quality than others, and we design our sampling distribution to over-sample documents from these systems. At the same time, given that our approach is a sampling-based approach the estimated evaluation measures are, by construction, unbiased on average, and assessments can be used to evaluate new, novel systems without introducing any systematic error. The method we propose, therefore, is an *active sampling* method with the probability distribution over documents changing in every round of assessments through the re-estimation of the quality of the participating runs. Accordingly, our solution consists of a *sampling* step and an *estimation* step. In the sampling step, we construct a distribution over runs and a distribution over documents in a ranked list and calculate a joint distribution over documents to sample from. In the estimation step, we use the Horvitz-Thompson estimator to correct for the bias in the sampling distribution and estimate evaluation measure values for all the runs. The estimated measures then dictate the new sampling distribution over systems, and hence a new joint distribution over the ranked collection of documents.

Therefore, the contribution of this chapter is a new sampling methodology for large-scale retrieval evaluation that combines the advantages of the sampling-based and the active-selection approaches. We demonstrate that the proposed method outperforms state-of-the-art methods in terms of effectiveness, efficiency, and reusability.

2.2 Active Sampling

In this section we introduce our new sampling method.

Table 2.1: Notation used throughout this chapter

Symbol	Description
$\mathcal{C}$	Depth-k document collection
S	Sample set
S'	Subset of S, only containing unique documents
N	Total number of unique documents in $\mathcal{C}$
K	Total number of contributing runs
T	Number of sampling rounds
N_b	Number of unique documents sampled in round t
N_t	Number of documents sampled in round t
d_i	i-th document
y_i	Relevance of document d_i
$r(i)$	Rank of document d_i
$p_t(k)$	Probability of k-th system run being sampled
$p_t(k, r(i))$	Probability of the document ranked r in k-th run being sampled

2.2.1 Active Sampling Algorithm

The key idea underlying our sampling strategy is to place a probability distribution over runs and a probability distribution over documents in the ranked lists of the runs, and iteratively sample documents from the joint distribution. In each round, we sample a set of documents from the joint probability distribution (batch sampling) and request relevance assessments by human assessors. The judged documents are then used to

update the probability distribution over runs. The process is repeated until we exhaust a predefined budget for human assessments (Figure 2.1).

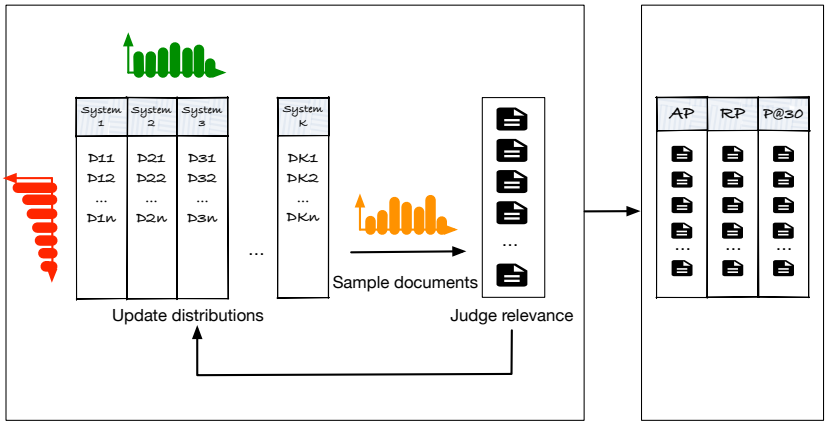


Figure 2.1: Active sampling and retrieval performance estimation.

Algorithm 1: Active sampling

Input: Prior distribution over runs $\{p_1(k)\}_{k=1}^{K}$, prior distributions over document ranks $\{p_1(k, r(i))\}_{i=1}^{N}$, document collection $\mathcal{C}$, batch size N_b.

Output: Sampled documents S, associated with relevance judgment and selection probability: $\{(d_{t,j}, y_{t,j}, p_t(j))\}_{j=1}^{N_b T}$.

for $t = 1, 2, \ldots, T$ **do**

- Calculate the joint document sampling distribution $p_t(i) = \sum_{k=1}^{K} p_t(k) p_t(k, r(i)), i = 1, \ldots, N$;
- Sample N_t documents with replacement from $p_t(i)$ so that it contains N_b unique documents;
- Let the sampled document be $d_{t,j}$; judge relevance of the sampled documents $y_{t,j}, j = 1, \ldots, N_t$;
- Augment data $S_{t+1} = S_t \cup \{(d_{t,j}, y_{t,j})\}_{j=1}^{N_t}$;
- Update distribution over runs $p_{t+1}(k), k = 1, \ldots, K$;

The notation used throughout the chapter is listed in Table 2.1. The process is illustrated in Algorithm 1. Initially, we provide a prior distribution over runs, denoted by $\{p_1(k)\}_{k=1}^{K}$, a prior distribution over the ranks of the documents, denoted by $\{p_1(k, r(i))\}_{i=1}^{N}$ for each run k , and the document collection $\mathcal{C}$. Given that we have no prior knowledge of the system quality, it is reasonable to use a uniform probability distribution over runs, i.e. $p_1(1) = p_1(2) = \ldots = p_1(K)$. In each round t,

we calculate the selection probabilities of the documents (that is the probability that a document is selected in each sampling round) $p_t(i)$ for each document i, and then sample a document on the basis of this distribution. We use *sampling with replacement with varying probabilities* to sample documents, which determines how we calculate the unbiased estimators and it is described in Section 2.3. The sampled documents $d_{t,j}$ are then judged by human assessors, with the relevance of these documents denoted as $y_{t,j}$, and the new data is added to S_t, which is used to update the $(t+1)$-th posterior distribution over runs.

2.2.2 Distribution over System Runs

The distribution over runs determines the probability of sampling documents from each run. Similar to active-selection methods, we make the assumption that good systems retrieve more relevant documents at the top of their rankings compared to bad systems. Based on this assumption we wish to over-sample from rankings of good systems.

Any distribution that assigns a higher probability to better performing systems could be used here. In this chapter we consider the estimated performance of the retrieval systems on the basis of the relevance assessments accumulated in each round of assessment as the system weights, and normalize these weights to obtain a probability distribution over runs. Different evaluation measures can be used to estimate the performance of each run after every sampling round.

Here we define a probability distribution proportional to the estimated average precision $\widehat{AP}$ introduced in Section 2.3.2:

$$p_t(k) = \frac{\widehat{AP_t}(k)}{\sum_{k=1}^{K} \widehat{AP_t}(k)}, k = 1, \ldots, K\,; t = 1, \ldots, T\,. \tag{2.1}$$

Figure 2.2 demonstrates the accuracy of the estimated (normalized) average precision at the end of four sampling rounds compared to the normalized average precision when the entire document collection (or to be more accurate the depth-100 pool for topic 251 in TREC 5) is used. In every round the estimates better approximates the target values (denoted with a line). The details of the measure approximations are provided in Section 2.3.

2.2.3 Distribution over Document Ranks

The distribution over document ranks for a system k determines the probability of sampling a document at a certain rank of the ordered list returned by run k. The underlying assumption that defines this probability distribution is that runs satisfy the probability ranking principle (PRP) [132] which dictates that the probability of relevance monotonically decreases with the rank of the document. Hence, if we let p denote the probability of sampling a document at rank r, then it is natural to assume p is a function of r and $p(r)$ monotonically decreases with r. Once again, any distribution that agrees with PRP can be used; previous researchers have used a number of such distributions (e.g., see [13, 70, 126]).

In this chapter we consider an AP-prior distribution proposed by Aslam et al. [13] and Pavlu et al. [126] which aims to define the probability at each rank on the basis of the contribution of this rank in the calculation of average precision. The intuition is that when rewriting $AP = \frac{1}{N}\sum_{1 \leqslant j \leqslant i \leqslant N} \frac{1}{i} y_i y_j$, the implicit weight associated with

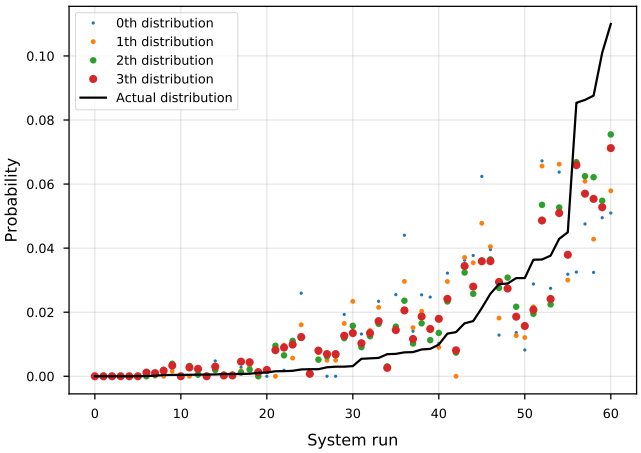


Figure 2.2: Probability distribution over runs on topic 251 in TREC 5. The black curve is the probability induced by the actual average precisions based on depth-100 pooling, while circular markers of different sizes denote the approximate probabilities for different runs. Runs have been sorted according to their actual average precision values.

rank r can be obtained by summing weights associated with all pairs involving r, i.e. $\frac{1}{N}\left(1+\frac{1}{r}+\frac{1}{r+1}+...+\frac{1}{N}\right)$. Then the AP-prior distribution is defined as follows:

$$w\left(r\right)=\frac{1}{N}\left(1+\frac{1}{r}+\frac{1}{r+1}+...+\frac{1}{N}\right)\approx\frac{1}{N}\log\frac{N}{r} \tag{2.2}$$

$$p\left(r\right)=\frac{w(r)}{\sum_{r=1}^{N}w(r)}\,, \tag{2.3}$$

where r is the rank of a document and N the total number of documents in the collection. Similar to Aslam et al. [13] and Pavlu et al. [126] and all other sampling-based methods, this distribution is defined at the beginning of the sampling process and remains fixed throughout the experiment.

2.3 Retrieval Effectiveness Estimator

In this section, we discuss the estimation of evaluation measures on the basis of the sampling procedure described in Algorithm 1. We first calculate the inclusion probabilities of each document in the collection, and then demonstrate how these probabilities can be used by a Horvitz-Thompson estimator to produce unbiased estimators of the population mean, and subsequently of some popular evaluation measures. The Horvitz-Thompson estimator, together with the calculated inclusion probabilities can be used to calculate the majority of the evaluation measures used in IR. Other measures can be derived in similar ways (e.g., see Table 1 in Schnabel et al. [142]).

2.3.1 Sampling Design

Sampling procedure. In each round of our iterative sampling process described in Algorithm 1, n documents are sampled from a collection of size N. In each round, the

unconditional probability of sampling a document d_i (*selection probability*) is $p_t(i)$, as defined in Step 2 of Algorithm 1, with

$$\sum_{i=1}^{N} p_t(i) = 1, p_t(i) \geq 0 \,; i = 1, 2, \ldots, N \,; t = 1, 2, \ldots, T \,. \tag{2.4}$$

Let i denote the index of the n documents composing the sample set. The probability of a document d_i being sampled (*first-order inclusion probability*) at the end of the sampling process is given by

$$\pi_i = 1 - \prod_{t=1}^{T} \prod_{z=1}^{N_t} (1 - p_t(i)) \,, \tag{2.5}$$

which accounts for varying probabilities across different rounds, while the probability of any two different document d_i and d_j being sampled (*second-order inclusion probability*) is given by

$$\pi_{ij} = \pi_i + \pi_j - \left[1 - \prod_{t=1}^{T} \prod_{z=1}^{N_t} (1 - p_t(i) - p_t(j)) \right] . \tag{2.6}$$

For the details of the derivation of the inclusion probabilities the reader can refer to Thompson [158]. Using these inclusion probabilities together with the Horvitz-Thompson estimator allows us to construct unbiased estimators for different evaluation measures in IR.

Horvitz-Thompson estimator of population total. Horvitz et al. [72] propose a sampling theory for constructing unbiased estimators of population totals. With any sampling design, with or without replacement, the unbiased Horvitz-Thompson estimator of the population total is

$$\widehat{\tau} = \sum_{i \in S'} \frac{y_i}{\pi_i} \,, \tag{2.7}$$

where S' is the subset of S that only contains unique documents.

An unbiased estimator of the variance of the population total estimator is given by

$$\widehat{var}(\mu) = \sum_{i \in S'} \left(\frac{1}{{\pi_i}^2} - \frac{1}{\pi_i} \right) {y_i}^2 + 2 \sum_{i > j \in S'} \left(\frac{1}{\pi_i \pi_j} - \frac{1}{\pi_{ij}} \right) y_i y_j \,. \tag{2.8}$$

For the details of these derivations the reader can refer to Thompson [158].

2.3.2 Evaluation Metrics

In this chapter we consider three of the most popular evaluation measures in IR, *precision at a certain cut-off r* ($PC(r)$), *average precision* (AP), and *precision at rank of R* (RP). We first clarify the exact expressions of the evaluation metrics with regard to the population, then introduce the estimators of these evaluation metrics on the sample set. Let $C = \{d_i\}_{i=1}^{N}$ denote a population of documents and let y_i be an indicator variable

of d_i, with $y_i = 1$ if the document d_i is relevant, and $y_i = 0$ otherwise. The *population total* is the summation of all y_i, i.e., the total number of relevant documents in the collection, while the *population mean* is the population total divided by the *population size*. If the population of documents considered is the set of documents ranked in the top-r for some run k, then the population mean is the precision at cut-off r.

Based on the definition, precision at cutoff r, average precision, and precision at rank R (R is the total number of relevant documents)are expressed as follows:

$$PC(r) = \frac{\sum_{d_i \in C, r(i) \leq r} y_i}{r} \tag{2.9}$$

$$AP = \frac{\sum_{d_i \in C} PC(r(i))\, y_i}{R} \tag{2.10}$$

$$RP = \frac{\sum_{d_i \in C, r(i) \leq R} y_i}{R}. \tag{2.11}$$

Suppose that we have sampled n documents $S = \{d_i\}_{i=1}^{n}$, with associated relevance labels $\{y_i\}_{i=1}^{n}$ and we wish to estimate R, $PC(r)$, AP and RP. The estimators for the four measures based on the Horvitz-Thompson estimator can be calculated by

$$\widehat{R} = \sum_{d_i \in S'} \frac{y_i}{\pi_i} \tag{2.12}$$

$$\widehat{PC}(r) = \frac{\sum_{d_i \in S', r(i) \leq r} \frac{y_i}{\pi_i}}{r} \tag{2.13}$$

$$\widehat{AP} = \frac{\sum_{d_i \in S'} \frac{\widehat{PC}(r(i)) y_i}{\pi_i}}{\widehat{R}} \tag{2.14}$$

$$\widehat{RP} = \frac{\sum_{d_i \in S', r(i) \leq \widehat{R}} \frac{y_i}{\pi_i}}{\widehat{R}}. \tag{2.15}$$

Note that both $\widehat{AP}$ and $\widehat{RP}$ are expressed as a ratio of two unbiased estimators. Although the numerator and denominator are unbiased (because $\widehat{R}$ and $\widehat{PC}(r)$ are unbiased), the ratio is not guaranteed to be unbiased. The bias tends to be small and decrease with an increasing sample size [127, 158].

2.4 Experimental Setup

In this section we introduce our research questions, the statistics we use to evaluate the performance of the proposed estimators, and the datasets and baselines used in our experiments. The implementation of the algorithm and the experimental runs are made publicly available.[1]

2.4.1 Research Questions

In the remainder of the chapter we aim to answer the following research questions:
RQ1 How does active sampling perform compared to other sampling-based and active-selection methods regarding bias and variance in the effectiveness measures?

[1] https://github.com/dli1/activesampling

RQ2 How fast do active sampling estimators approximate the actual evaluation measures compared to other sampling-based and active-selection methods?
RQ3 Is the test collection generated by active sampling reusable for new runs that do not contribute in the construction of the collection?
The aforementioned questions allow us to have a thorough examination of the effectiveness as well as the robustness of the proposed method.

2.4.2 Statistics

To answer the research questions put forward in the previous section, we need to quantify the performance of different document selection methods.

Our first goal is to measure how close the estimation of an evaluation measure is to its actual value when the full judgment set is being used. Assume that a document selection algorithm chooses a set of documents S to calculate an evaluation measure. Let us denote the estimated evaluation measure, calculated on the judgment sample S for some run k, by $f(k \mid S)$. Let us also assume that the actual value of that evaluation measure, when the full judgment set is used, is $h(k)$. Following the work of Pavlu et al. [126], we use both the *root mean squared error (rms)* and the *mean squared error (mse)* error of the estimator over a sample set, which measure how close on average the estimated and the actual values are. They are expressed as follows:

$$rms = \mathbb{E}_S \sqrt{\mathbb{E}_k (f(k \mid S) - h(k))^2} \tag{2.16}$$

$$mse = \mathbb{E}_S \mathbb{E}_k (f(k \mid S) - h(k))^2 . \tag{2.17}$$

To further decompose the estimation errors made by different methods we also calculate the *bias* and the *variance* by decomposing the *mse*. *Bias* expresses the extent to which the average estimator over all sample sets differs from the actual value of a measure, while *variance* expresses the extent to which the estimator is sensitive to the particular choice of a sample set (see [21]). According to Bishop [21], *mse* can be further rewritten as

$$\begin{aligned} mse &= \mathbb{E}_S \mathbb{E}_k (f(k \mid S) - h(k))^2 \\ &= \mathbb{E}_k \mathbb{E}_S (f(k \mid S) - h(k))^2 \\ &= \mathbb{E}_k \left((\mathbb{E}_S (f(k \mid S) - h(k)))^2 + \mathbb{VAR}_S f(k \mid S) \right) \\ &= \mathbb{E}_k \mathbb{E}_S (f(k \mid S) - h(k)) + \mathbb{E}_k \mathbb{VAR}_S f(k \mid S) \\ &\triangleq bias + variance . \end{aligned} \tag{2.18}$$

A second measurement we are interested in is how far the inferred ranking of systems when estimating an evaluation measure is to the actual ranking of systems when the entire judged collection is being used. Following previous work [10, 11, 126, 172, 174], we also report the Kendall's τ between estimated and actual rankings. Even though the Kendall's τ is an important measure when it comes to comparative evaluation, *rms* error remains our focus, since test collections have found use not only in the evaluation of retrieval systems but also in learning retrieval functions [101]. In the latter case, for some algorithms, the accuracy of the estimated values is more important than just the correct ordering of systems.

2.4.3 Test Collections

We conduct our experiments on the TREC 5–8 AdHoc and TREC 9–11 Web tracks. The details of the datasets can be found in Table 2.2. In our experiments we did not exclude any participating run, and we considered the relevance assessments released by NIST as the complete set of judgments over which the actual values of measures are being computed.

Table 2.2: Test collections.

TREC	Task	Topics	# runs	# rel doc	# qrel	# rel doc per query	# qrel per query
TREC-5	Adhoc	251-300	61	5524	133681	110.48	2673.6
TREC 6	Adhoc	301-350	74	4611	72270	92.22	1445.4
TREC 7	Adhoc	351-400	103	4674	80345	93.48	1606.9
TREC 8	Adhoc	401-450	129	4728	86830	94.56	1736.6
TREC 9	Web	451-500	104	2617	70070	52.34	1401.4
TREC 10	Web	501-550	97	3363	70400	67.26	1408.0
TREC 11	Web	551-600	69	1574	56650	31.48	1133.0

2.4.4 Baselines

We use two active-selection and one sampling-based methods as baselines:

Move-to-front (MTF) [43] MTF is a deterministic, iterative selection method. In the first round, all runs are given equal priorities. In each round, the method selects the run with the highest priority and obtains the judgment of the first unassessed document in the ranked list of the given run. If the document is relevant the method selects the next unassessed document until a non-relevant document is judged. If that happens, the priority of the current run is being reduced and the run with the highest priority is selected next.

Multi-armed bandits (MAB) [109] Similar to MTF, MAB aims to find as many relevant documents as possible. MAB casts document selection as a multi-armed bandit problem, and different to MTF it randomly decides whether to select documents from the best run in the current stage, or sample a document across the entire collection. For the MAB baseline we used the best method MM-NS with its default setting reported in [109].[2]

Stratified sampling (Stratif) [126] Stratif is a stochastic method based on *importance sampling*. The probability distribution over documents used is the AP-prior distribution, which remains unchanged throughout the sampling process. Similar to our approach, the Horvitz-Thompson estimator is used to estimate the evaluation metrics. The stratified sampling approach proposed by Pavlu et al. [126] has been used in the construction of the TREC million query track collection [27]; it outperforms methods using uniform random sampling [10, 172] and demonstrates similar performance to the method by Yilmaz et al. [174].

[2] `http://tec.citius.usc.es/ir/code/pooling_bandits.html`

2.5 Results and Analysis

2.5.1 Bias and Variance

This first experiment is designed to answer RQ1 and is conducted on TREC 5.

We reduce the retrieved document lists of all runs to the top-100 ranks (so that all documents in the ranked lists are judged) and consider this the ground truth rankings, based on which the actual values of MAP, RP and $P@30$ are calculated. The judgment effort is set to 10% of the depth-100 pool for each query, and different methods are used to obtain the corresponding subset and calculate the estimated MAP, RP and $P@30$ for each run. For any stochastic method (i.e., the sampling methods and MAB) the experiment is repeated 30 times. Based on the estimated and actual values we calculate mse, and its decomposition to $bias$ and $variance$ for each estimator. The batch size N_b for all the experiments has been set to 3 throughout the chapter.

Figure 2.3 shows a number of scatter plots for MTF, MAB, Stratif, and our method denoted as Active. Each point in the plots corresponds to a given run. To declutter the figure, the shown points for the sampling-based methods are computed over a single sample. An unbiased estimator should lead to points that lie on the $x = y$ line. As it can be observed the active sampling estimated values are the ones that are closer to the diagonal. As expected, and by construction, precision is unbiased, while the bias introduced in the ratio estimators of AP and RP is smaller that all active-selection methods, and comparable to the stratified sampling method.

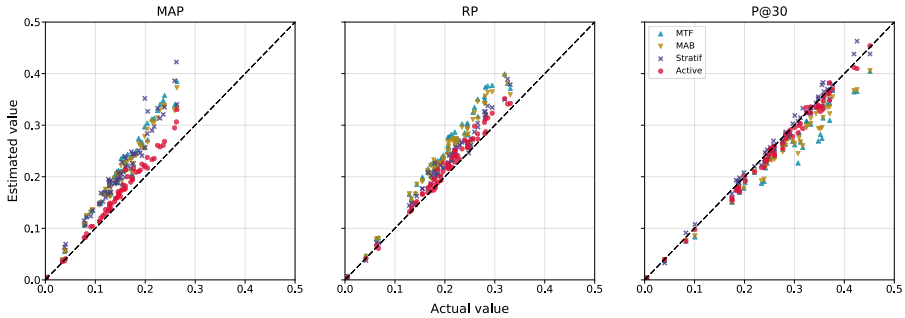


Figure 2.3: Estimated vs actual values of MAP, RP and $P@30$ for different runs on a 10% sample of TREC 5.

A decomposition of the mse into $bias$ and $variance$ can be found in Figure 2.4. As expected, the variance of active-selection method is zero (or close to zero) since MTF is a deterministic method, while the randomness of MAB is only in the decision between exploration and exploitation. Active sampling has a much lower variance than stratified sampling, which demonstrates one of the main contribution of our sampling method: biasing the sampling distribution towards good performing runs improves the estimation of the evaluation measures. The bias of the sampling-based methods, as expected, is near-zero, while it is larger than zero for the active-selection methods, since they do not correct for their preference to select documents from good performing runs. For example, the bias on $P@30$ of active-selection methods are much smaller than zero, because the greedy strategies only count the number of relevant documents

and thus underestimate $P@30$; while the sampling methods can avoid the problem by using unbiased estimators. This demonstrates the second main contribution of our approach: using sampling avoids any systematic error in the calculation of measures. Therefore, the proposed sampling method indeed combines the advantages of both sampling-based and active-selection methods that have been proposed in the literature.

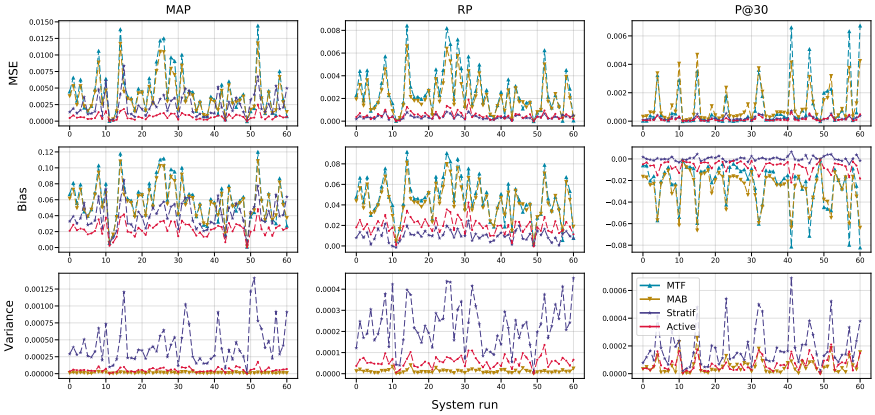


Figure 2.4: The mse error , $variance$, and $bias$ (y-axis) of the sampling-based and active-selection methods compared, for different runs (x-axis) over 30 sample sets drawn from TREC 5. The *mse* of Active is significantly smaller than that of MTF/MAB/Stratif for MAP/RP/$P@30$ at 95% confidence level by Welch's t-test, except for Stratif on RP where Active is significantly larger than Stratif.

2.5.2 Effectiveness

Our second experiment is designed to answer RQ2 and is conducted on TREC 5–11. In this experiment we vary the judgment effort from 1% to 20% of the depth-100 pool. At each sampling percentage, when sampling-based methods are used, we first calculate the rms error and Kendall's τ values for a given sample and then average these values over 30 sample sets.

Figure 2.5 shows the average rms and τ value at different sample sizes. For all TREC tracks active sampling demonstrates a lower rms error than stratified sampling, MTF, and MAB for sampling rates greater than 3–5%. At lower sampling rates active-selection methods show an advantage compared to sampling-based methods that suffer from high variance. Regarding Kendall's τ, active sampling outperforms all methods for TREC 5–8, for sampling rates greater than 5%, while for TREC 10 and 11 it picks up at sampling rates greater than 10%. TREC 10 and 11 are the two collections with the smallest number of relevant documents per query, hence finding these document using active-selection methods leads to a satisfactory ordering of systems when the percentage of judged documents is very small. For those small percentages the sampling-based methods demonstrate high variance, and it really depends on how lucky one is when drawing the sample of documents. The variance of rms error and Kendall's τ across the 30 different samples drawn in this experiment for the estimation of MAP on TREC 11 can be seen in Figure 2.6.

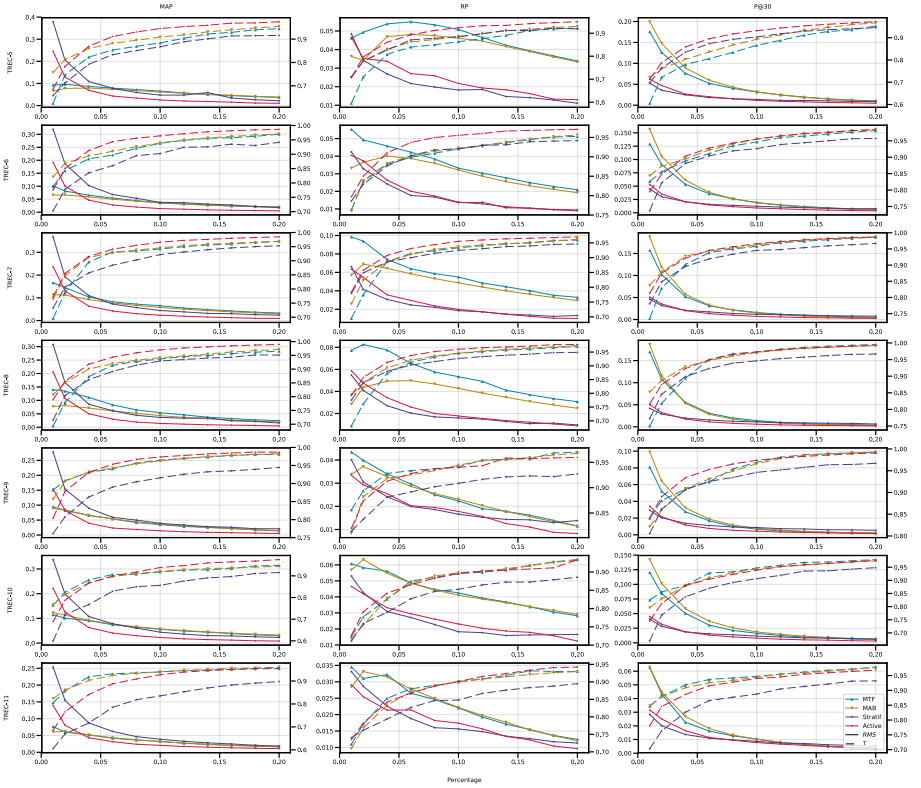


Figure 2.5: Average rms and τ for the sampling-based and active-selection methods at different sample sizes in TREC 5–11. The left y-axis and solid lines denote rms, the right y-axis and dotted lines denote Kendall's τ.

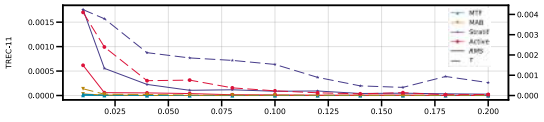


Figure 2.6: Variance of rms error (solid lines) and Kendall's τ values (dashed lines) when estimating MAP, over 30 sample for different sample sizes TREC 11.

Overall, when comparing Active with MTF and MAB, we find that our method outperforms them regarding rms. This indicates once again that the calculated inclusion probabilities and the Horvitz-Thompson estimator allow Active to produce an unbiased estimation of the actual value of the evaluation measures. When comparing Active with Stratif, both of which use the Horvitz-Thompson estimator, we can find that our method outperforms stratified sampling in terms of Kendall's τ. This indicates that the dynamic strategy we employ is beneficial compared to a static sampling distribution. Therefore, our active sampling method indeed combines the advantages of both methods.

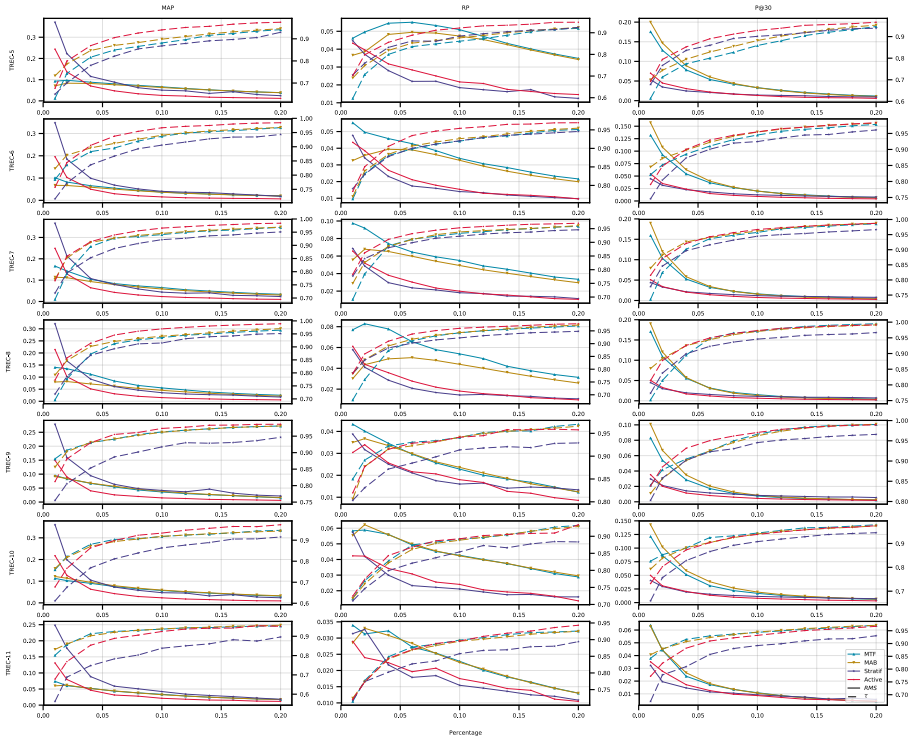


Figure 2.7: Average rms and τ of the sampling-based and active-selection methods at different sample sizes on leave-one-group-out runs in TREC 5–11.

2.5.3 Reusability

Constructing a test collection is a laborious task, hence it is very important that the proposed document selection methods construct test collections that can be used to evaluate new, novel algorithms without introducing any systematic errors. This experiment is designed to answer RQ3 and is conducted on TREC 5–11. In this experiment we split the runs into contributing runs and left-out runs. Using the contributing runs we construct a test collection for each different document selection method. We then calculate the estimated measures for all runs including those that were left out from the collection construction experiment. In our experiment, we use a one-group-out split of

the runs. Runs that contributed in the sampling procedure come from different participating groups. Groups often submit different versions of the same retrieval algorithm, hence, typically, all the runs submitted by the same participating group differ very little in the ranking of the documents. To ensure that left-out runs are as novel as possible we leave out all runs for a given group. Regarding the calculation of the rms error and Kendall's τ we compute rms error and Kendall's τ considering both participating and left-out runs.

Figure 2.7 shows the average rms error and Kendall's τ values at different sample sizes using the latter afore-described option to isolate the effect of the different document selection methods on new, novel systems. In general, the trends observed in Figure 2.5 can also be observed in Figure 2.7, with active sampling outperforming all other methods regarding rms error and Kendall's τ for sampling rates greater than 5%. For sampling rates lower than 5% in collections with very few relevant documents per topic (such as TREC 10 and 11) the active-selection methods perform better than the sampling-based methods, however we can also conclude that at these low sampling rates none of the methods lead to reliably reusable collections.

2.6 Conclusion

In this chapter, we have answered **RQ1** by devising a sampling-based approach – *active sampling*. Our method consists of a sampling step and an unbiased estimation step. In the sampling step, we construct two distributions, one over retrieval systems that is updated in every round of relevance assessments giving larger probabilities to better quality runs, and one over document ranks that is defined at the beginning of the sampling process and remains static throughout the experiment. Document samples are drawn from the joint probability distribution, and inclusion probabilities are computed at the end of the entire sampling process, accounting for varying probabilities across sampling rounds. In the estimation step, we use the well-known Horvitz-Thompson estimator to estimate evaluation metrics for all system runs.

The proposed method is designed to combine the advantages of two families of methods that have appeared in the literature: sampling-based and active-selection approaches. Similar to the former, by construction, our method leads to unbiased, estimators of evaluation measures, and can safely be used to evaluate new, novel runs that have not contributed to the generation of the test collection. Similar to the latter, the attention of our method is put on good quality runs with the hope of identifying more relevant documents and reduce the variability naturally introduced in the estimation of a measure due to sampling.

To examine the performance of the proposed method, we tested against state-of-the-art sampling-based and active-selection methods over seven TREC ad-hoc and web collections, TREC 5–11. Compared to sampling-based approaches, such as stratified sampling, out method indeed demonstrated lower variance, while compared against active-selection approaches, such as the move-to-front method and the multi-armed bandits method, our method, as expected, has lower, near-zero bias. For sampling rates as low as 5% of the entire depth-100 pool, the proposed method outperforms all other methods regarding effectiveness and efficiency and leads to reusable test collections.

In the next chapter, we study the automatic thresholding problem of document

selection. We further take the cost of assessing documents into consideration, hoping to make the document selection process in an effective manner to maximize recall and minimize assessment cost and in a transparent manner so that we know the number of residual relevant documents.

3 Automatic Thresholding of Document Selection

In Chapter 2, we have studied how to actively select documents for the construction of a test collection and the evaluation of retrieval systems. In this chapter, we further take relevance assessment cost into consideration. We study how to determine the stopping point of document selection for constructing test collections that balance the cost of assessing document relevance and the gain of identifying relevant documents. This chapter aims to answer the following research question asked in Chapter 1:

RQ2 How can we select documents for assessing relevance and stopping the selection, both in an effective manner to maximize recall and minimize assessment cost and in a transparent manner so that we know the number of residual relevant documents?

3.1 Introduction

Given an information need, exhaustive search aims to retrieve all the relevant information, if possible. There is a real demand for exhaustive search in domains like electronic discovery, systematic review, investigation, research, or even the construction of datasets for information retrieval evaluation [41, 42, 60, 81, 83, 85, 94, 113, 135]. Electronic discovery involves searching electronic business records such as correspondence, memos, emails, and balance sheets for documents that are relevant or responsive to a lawsuit or a government investigation. It is an important aspect of the civil litigation process in the United States [123]. Missing relevant business records may cause significant impact on the lawsuit or the government investigation. Systematic review is a type of literature review that uses systematic methods to reliably bring together the findings from multiple studies that address a question and are often used to inform policy and practice, e.g., the development of medical guidelines in evidence-based medicine [124]. Test collections are fundamental in IR evaluation. Assessing large-scale relevance labels is inevitable in order to build a high quality test collection. Missing relevant labels in test collections causes bias when evaluating retrieval models [42]. The electronic business records, studies, or documents in these domains are usually large and their number is growing rapidly, making the task of identifying all relevant information both complex and time consuming. The technology-assisted review (TAR) approach tackles the problem by using classification or ranking algorithms

to identify the relevant documents based on relevance feedback from expert reviewers, until a substantial number (or all) of the relevant documents have been identified.

The continuous active learning (CAL) approach and its extensions have demonstrated high effectiveness when used in the TAR process [35, 36, 38, 39]. Given a document collection and a query, a ranker (or a classifier) is trained to identify documents to be shown to expert reviewers for relevance assessment. Then, the assessed documents are used as training data to re-train the ranker. As more and more documents are identified by the ranker and assessed by the reviewers, the training data grows and the performance of the ranker improves. The TAR process continues until "enough" relevant documents have been found. "Enough" is often specified by the additional cost (in terms of reading irrelevant documents) required to find more relevant documents, or by the importance of the missing relevant documents, e.g., towards resolving a legal dispute [38], or even as a percentage of relevant documents. In this chapter, we assume the latter, i.e., the experts specify a desired *recall level* to determine when should the reviewing process end.

In this case the goal of the ranker is two-fold: first, to identify relevant documents as early as possible during the TAR process and second, to accurately estimate the total number of relevant documents, R, in the collection so that we can stop the TAR process in a transparent manner, when we reach the required level of recall.

The two goals are, to some extent, conflicting in terms of the optimal strategy of stopping the TAR process. To demonstrate this, consider the following example. Suppose we have an unknown perfect ranking model that ranks all the relevant documents before the non-relevant documents. The most effective strategy in terms of achieving high recall is to select documents starting from the top of the list and moving towards the bottom. However, in order to know the total number of relevant documents in the collection one needs to examine all the documents in the collection. On the other hand, a random sampling design[1] can produce an unbiased estimator of the total number of relevant documents, R, by selecting a relatively low number of documents to review, leading however to low recall, if the sampled documents are the only ones the reviewer reads.

There is little work that tackles the problem of automatically stopping the TAR process and they follow two directions. The first direction [35, 50, 110] proposes different methods to determine an ad-hoc stopping point. However, such approaches lack transparency because they provide no information on how many relevant documents are still not found. The second direction [37, 165] first samples documents to obtain an unbiased estimate of R, paying a significant assessment cost, and then employs a TAR method to rank and find the required number of relevant documents. None of the directions that we just described can both produce effective rankings and support transparently the decision to stop reviewing documents.

In this chapter, we tackle the problem of determining the stopping point of document selection for constructing test collections that balance the cost of assessing document relevance and the gain of identifying relevant documents, by proposing a novel framework for the TAR process. The framework consists of a ranking module, a

[1] A *sampling design* contains a sampling distribution, the manner to sample documents from the distribution (e.g., with replacement or without replacement), and some statistical estimators of desired values such as population total [158].

sampling module, an assessment module, an estimation module, and a stopping module (see Figure 3.1 and Section 3.3.1). The ranking module provides a ranked list of documents based on their predicted relevance. The sampling module consists of a sampling distribution and a manner to sample documents from the distribution. The estimation module provides an estimator of R as well as its variance based on the sampling distribution and the sampled documents. The stopping module determines whether to stop the TAR process. Our framework approaches TAR in an active learning manner following the CAL approach [35]. The major difference from the CAL approach is that we allow random sampling from all the documents in the collection instead of greedily assessing documents from top to bottom, in a with-replacement sampling design that can both collect many relevant documents (Section 3.3.2) and produce an unbiased estimator of R with low variance (Section 3.3.3). Further, we devise different stopping strategies based on the estimated R and its variance (Section 3.3.5).

To summarize, in this chapter we make the following contributions:

- We propose a novel framework for the TAR process that allows us to conduct a "greedy" sampling over documents in order to collect as many relevant documents as possible, and produce a sequence of statistically unbiased estimators of R.
- We provide a proof of the unbiasedness of the proposed estimators of R under our sampling design and also empirically verify the unbiasedness.
- We develop three datasets for the task of total recall, including the CLEF Technology-Assisted Reviews in Empirical Medicine datasets in 2017, 2018 and 2019, respectively [81, 83, 85].
- We validate the effectiveness of the proposed framework and provide a detailed analysis of its components on various datasets including the CLEF Technology-Assisted Reviews in Empirical Medicine datasets [81, 83, 85], the TREC Total Recall datasets [60], and the TREC Legal datasets [41].
- We reproduce a large number of baselines, and we release the code, along with the code of our work.

3.2 Related Work

In this section we first introduce the CAL approaches since our framework is developed based on these approaches. Then we compare the merits and drawbacks of different methods that tackle the problem of stopping the TAR process. Finally, we explain the main differences between our method and existing work.

3.2.1 Continuous Active Learning Approaches

The TAR process aims to iteratively retrieve a substantial number (if not all) of the relevant documents in a collection – which makes it a *total recall* problem. Cormack and Grossman proposed a family of CAL approaches used in various total recall tasks, including technology-assisted reviews in electronic discovery and in empirical medicine, as well as the construction of test collections in information retrieval [35, 38]. The first autonomous TAR (AutoTAR) method proposed by Cormack et al. [38] mainly consisted of an initial selection of training documents by a simple keyword search, and subsequent selections by active learning. The method significantly outperformed simple active learning with uncertainty sampling in terms of human reviewing cost.

Cormack et al. [35] later proposed a different instantiation of AutoTAR, which enhances the first CAL method through a handful of adaptions including TF-IDF features, a single relevant seed document, pseudo non-relevant documents, and exponentially increasing batch size of documents to be reviewed at each iteration. AutoTAR is considered the current state-of-the-art method for total recall tasks.

However, AutoTAR leaves the question of when to terminate the reviewing process open. In practice, there is no way to know the number of relevant documents in a collection before inspecting and labelling every document. It is thus impossible to know exactly what level of recall has been reached during the process. Hence, one is facing a dilemma between high recall and low cost in terms of reviewing documents. Stopping reviewing documents too early will result in missing many valuable relevant documents; stopping too late will cause unnecessary cost when there are no more relevant documents to be found.

So far, researchers have developed various approaches to solve the "stopping" problem. We introduce them in the following two sections.

3.2.2 Topic-wise Approaches

In their AutoTAR experiment, Cormack et al. [35] observed that the *gain curve* (i.e., recall as a function of the number of documents reviewed) shows clearly diminishing returns at some point, and that if a substantial number of relevant documents have been found with high precision and precision drops later on, the vast majority of relevant documents have most likely been found. Inspired by this observation, Cormack et al. [37] proposed the *Knee* method, the *Target* method, and the *Budget* method. The *Knee* method defines a *knee* of the gain curve through a simple geometric algorithm [140]. The TAR process stops when the slope after the knee diminishes to less than a ratio (e.g., $\frac{1}{6}$) of the slope before the knee. In the *Target* method Cormack et al. [37] first reviews a randomly sampled set of documents, until a pre-defined number (e.g., 10) of them are judged relevant, which is the target set. The documents in the collection are ranked and retrieved within the AutoTAR framework without knowledge of the target set, until each document in the target set has been retrieved. In the *Budget* method Cormack et al. [37] combine the *Knee* and the *Target* method. The TAR process stops when all the documents in the target set have been retrieved and the slope after the knee diminishes to less than a ratio (e.g., $\frac{1}{6}$) of the slope before the knee.

It has been shown empirically that the *Knee* method is the most effective one [37]. However the *Knee* method is a "blind" method and it does not indicate how many relevant documents have not been found.

To solve the aforementioned problem, Cormack et al. [39] proposed SCAL, which is designed to achieve high recall for a large scale or infinite document collection. In the first step, SCAL randomly samples a large subset of documents from the document collection in order to have an accurate $\widehat{R}$ – the estimator of R. *Stratified sampling* is applied to the subset to calculate $\widehat{R}$: it first splits the subset into many small strata, and then randomly samples documents from each stratum to estimate the total number of relevant documents in each stratum, finally $\widehat{R}$ is calculated by summing up the total number of relevant documents over all the strata and multiplying it by a calibration factor. At each iteration, a ranking model is also trained by using the sampled labeled documents as the training data. In the second step, $\widehat{R}$ is used to define a threshold with

which SCAL can select a ranking model from the sequence of ranking models and produce a ranked list of documents for the reviewer to review.

Wallace et al. [165] proposed several estimators of R and let reviewers decide when to stop by showing them how close they are to $\widehat{R}$. However, there is no guarantee that the estimators they proposed are statistically unbiased.

Di Nunzio [50] proposed a thresholding method by investigating the interaction between the probability of observing document d given the current relevant documents – $p(d \mid \mathcal{R})$ and the same probability given the non-relevant documents – $p(d \mid \mathcal{NR})$. First, a document d is represented as two coordinates $(p(d \mid \mathcal{NR}), p(d \mid \mathcal{R}))$, denoted by (x, y). Then the original problem of classifying (or ranking) documents can be transformed into an intuitive geometric problem of finding two decision lines in the form of $y = \alpha x + \beta$: (1) a line $y = \alpha x + \beta_{rel}$ that fits the existing assessed relevant documents, and (2) a parallel line $y = \alpha x + \beta_{least}$ passing through the least relevant documents. The TAR process stops when all the documents between the two lines are assessed.

Modelling the distribution of relevant and non-relevant documents and fitting them over scores in a reasonable way, which is called *score distribution* method [15, 87, 168], has been studied since the early days of information retrieval and is beneficial to a wide range of applications, as well as the "stopping" problem [6, 133]. Hollmann et al. [71] proposed a thresholding method based on score distributions. It first fits a Gaussian distribution to the scores of relevant articles from the relevance feedback of random sampled articles, and then estimates the total number of relevant documents at any rank position.

3.2.3 Cross-topic Approaches

Losada et al. [110] proposed a stopping method for accurate evaluation of retrieval systems using partially labeled test collections, with the estimation of R being a byproduct. During the training phase, a document pool which consists of multiple ranked lists of documents from different retrieval systems is formed; a power law function is used to model the number of new relevant documents at each pool depth; the number of relevant documents is also know; during the test phase, for a test query, a certain number of documents are assessed based on which the similarity between the pattern of relevance of the current test query and the pattern of relevance of each training query is calculated; finally, R is estimated by averaging the number of relevant documents of each training query weighted by their similarity.

Although the work of Losada et al. [110] studies the stopping problem, its goal is to achieve an accurate evaluation of retrieval systems, and not high recall per se. Besides, it assumes the existence of different retrieval systems, which produce multiple ranked lists. Pooling these systems is critical since it is then that the number of new relevant documents at each pool depth follows a power law distribution, based on which the method is designed. The method could be adapted, reducing the pool of ranking systems to a single ranking, and the pool of documents to the top-k documents of that ranking. Despite the fact that such a method could be used towards stopping the reviewing process, such an adaptation goes beyond the intentions of Losada et al. [110].

3.2.4 Summary

Inspired by Cormack et al. [35, 37, 39], we propose to jointly estimate R and improve the ranker at each iteration by "greedily" sampling documents. Our method is able to decide when to stop the TAR process with transparency – by estimating how many relevant documents are still missing. Such a model has several merits. First, it is topic-wise independent, which means no extra training topics are needed. Second, it does not not need an extra procedure to estimate R; instead, it iteratively utilizes the sampling module to collect relevant documents as well as to estimate R. Third, instead of estimating R once, it produces $\widehat{R}$ at every iteration, compensating for variance and reducing the risk of very wrong estimations. Fourth, it calculates $\widehat{R}$ as well as the variance estimator of $\widehat{R}$, enabling $\widehat{var}(\widehat{R})$ to also contribute to the stopping decision.

3.3 Method

In this section, we first introduce the overall framework, then we elaborate on the sub-modules. The notation used throughout this section is summarized in Table 3.1.

Table 3.1: Notation used in Section 3.3.

Symbol	Description
q	The input topic
$\mathcal{C}_q$	Document collection for topic q
N	Total number of documents in $\mathcal{C}_q$
R	Total number of relevant documents in $\mathcal{C}_q$
S	Set of sampled documents
n	Total number of documents in S
t	Iteration number
b_t	Number of documents to be sampled at t-th iteration
k	Number of pseudo relevant documents added per iteration
$\mathcal{L}_t$	Labelled documents (training set) at t-th iteration
$\mathcal{U}_t$	Unlabeled documents at t-th iteration
d_i^t	Document indexed by i at t-th iteration
r_i^t	Rank of d_i^t
y_i^t	Relevance label of d_i^t
p_i^t	Selection probability of d_i^t
π_i	First-order inclusion probability of d_i
$\pi_{i,j}$	Second-order inclusion probability of d_i and d_j
$'$	Operation of removing duplicates
$\sim$	Operation of cumulating units in previous iterations
$\widehat{\ }$	Estimator of a variable or statistic

3.3.1 The Framework

We propose a novel framework for the TAR process and we call it *autostop*. The framework mainly consists of a ranking module, a sampling module, an assessment

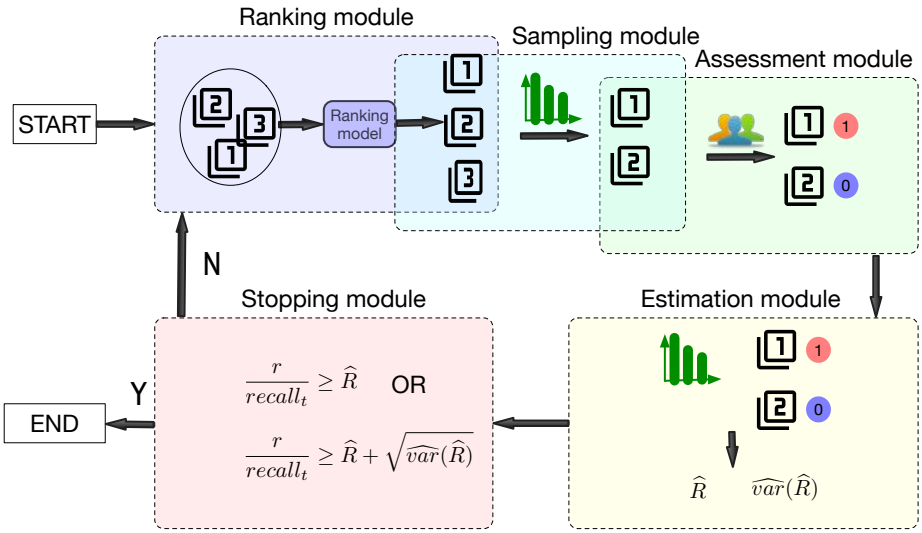


Figure 3.1: The proposed framework for the TAR process.

module, an estimation module and a stopping module (see Figure 3.1).

Given a topic a reviewer is interested in – denoted by q – a collection of documents for the topic – denoted by $\mathcal{C}_q$ – the reviewer can specify a target recall level – denoted by $recall_t$, which indicates what proportion of relevant documents retrieved by the framework makes the reviewer satisfied. The framework iteratively trains a ranking model to produce a ranked list of documents from which it samples documents for the reviewer to read towards satisfying her or his information need; then the framework outputs an estimator of the total number of relevant documents of the collection, and an estimator of the variance of the estimator. The process stops once the estimated recall exceeds the target recall.

We describe the framework in Algorithm 2. We start with the ranking and the sampling process. A ranking model will be trained from scratch and produce a ranked list for the documents in $\mathcal{C}_q$. Let $\mathcal{L}_t$ denote the (labeled) training set for the ranking model at iteration t, and $\mathcal{U}_t$ denote the unlabeled set at iteration t. Initially, $\mathcal{L}_0$ is empty and we fill it with a pseudo document d_0. We follow the AutoTAR method [35] to construct the pseudo document, i.e., we use the description of the topic. For the cases where there is no description for the topic, one can always use the query expanded by relevant feedback. Next we also add non-relevant documents into the training set. At each iteration, $\mathcal{L}_t$ is augmented with k documents, which are sampled uniformly and without replacement from $\mathcal{U}_t$ (in line 3). These documents are temporarily labeled non-relevant, same as the AutoTAR method [35]. In line 4, a ranking model is trained on $\mathcal{L}_t$. In line 5, the ranking model predicts the probability of relevance over all the documents in $\mathcal{L}_t$ and $\mathcal{U}_t$ (which equals $\mathcal{C}_q$). These documents are ranked in the order of decreasing relevance probability. In line 6, a sampling distribution $\mathcal{P}_t$ is constructed based on the ranked list of documents. We propose to use the AP Prior distribution and provide the details in Section 3.3.2. In line 7, a number of b_t documents are sampled independently and with replacement from $\mathcal{P}_t$.

After the documents are sampled, we start the assessment process. In line 8, the reviewer assesses the relevance of the sampled documents. Note that the sampled b_t documents may contain duplicates, therefore the reviewer only needs to assess the unique documents. Moreover, the sampling design is with-replacement, therefore it is possible to sample documents which have been assessed before. As a consequence, the reviewer only assesses at most b_t documents. In line 9, we follow the same method with the AutoTAR method [35], i.e., we remove the temporary documents from $\mathcal{L}_t$. Meanwhile, we add the b_t assessed documents (which may contain duplicates) in $\mathcal{L}_t$ and remove them from $\mathcal{U}_t$.

Given the sampled documents, their relevance labels, and their sampling probabilities, we can estimate the total number of relevant documents – denoted by $\widehat{R_t}$, as well as the variance of $\widehat{R_t}$ – denoted by $\widehat{var}(\widehat{R_t})$. They are calculated in line 10. We propose to use two estimators for R_t, which are the Horvitz-Thompson estimator and the Hansen-Hurwitz estimator. We provide the details in Section 3.3.3.

Finally, the stopping module uses $\widehat{R_t}$ and $\widehat{var}(\widehat{R_t})$ to decide whether to stop the TAR process or not. We propose two stopping strategies, an *optimistic* and a *conservative* one, and we explain them in detail in Section 3.3.5.

3.3.2 Sampling Design

In this section we elaborate on our sampling design with a focus on the sampling distributions. Note that in Algorithm 2 we need to sample documents from a distribution $\mathcal{P}_t = \{p_i^t\}$ (for notation simplicity we omit t and use $\mathcal{P} = \{p_i\}$). Ideally, p_i should be positively correlated with the relevance label at position i, which allows an estimator $\widehat{R}$ with low variance (see the explanation in Section 3.3.3). However, the relevance labels are not known until documents are assessed by the reviewers. What we have instead is a ranking model that can predict relevance probability and output a list of ranked documents. Therefore we construct $\mathcal{P}$ based on the output ranked list of documents.

The probability over document rank is defined as the probability of sampling a document at a certain rank of the ranked list. The underlying assumption is that a ranking model satisfies the PRP [132] which dictates that the probability of relevance monotonically decreases with the rank of the document. Any distribution that agrees with PRP can be used.

When choosing the sampling distribution, we are faced with a tradeoff between collecting as many relevant documents as possible and an accurate estimator of R with low variance. There are many ways to construct a distribution to sample from, for example, sampling from the output probability of relevance produced by the trained ranking model, or sampling from a power law distribution produced based on document ranks. We tried both and neither performed well enough. To meet the goal, we propose to use AP-prior distribution. It is a widely used sampling distribution in the task of information retrieval evaluation and it has been empirically proved to be a good prior for the relevance of documents in a ranked list [9, 10, 12, 13, 94, 173, 175].

AP-prior Distribution. The AP-prior distribution is proposed in [13, 126], which defines the probability at each rank position on the basis of the contribution of this rank position in the calculation of average precision. The idea is that we first rewrite *average precision* as $AP = \frac{1}{N}\sum_{1 \leqslant j \leqslant i \leqslant N} \frac{1}{i} y_i y_j$, where i, j denote the position in the ranked

Algorithm 2: Automatic thresholding algorithm

Input: Topic q; document collection $\mathcal{C}_q$, target recall $recall_t$.
Output: A list of labeled documents $\{(d_i^t, y_i^t) \mid i = 1, 2, \ldots; t = 1, 2, \ldots\}$; a list of estimator $\{\widehat{R_t} \mid t = 1, 2, \ldots\}$ and $\{\widehat{var}(\widehat{R_t}) \mid t = 1, 2, \ldots\}$.

t = 0, $\mathcal{L}_0 = \{$pseudo document $d_0\}$;
while not stop **do**
    t += 1;
    `// Sampling`
    Temporarily augment $\mathcal{L}_t$ by uniformly sampling k documents from $\mathcal{U}_t$, labeled non-relevant;
    Train a ranking model on $\mathcal{L}_t$;
    Rank all the documents in $\mathcal{C}_q$ with the ranker trained over $\mathcal{L}_t$;
    Construct sampling distribution $\mathcal{P}_t$ over the ranked documents via Equation (3.1);
    Sample b_t documents from the distribution $\mathcal{P}_t$;
    Render relevance assessments for the sampled documents;
    Remove the k temporary documents from $\mathcal{L}_t$;
    Place the b_t assessed documents in $\mathcal{L}_t$, and remove them from $\mathcal{U}_t$;
    $b_{t+1} = b_t + [\frac{b_t}{10}]$;
    `// Estimation`
    Calculate $\widehat{R_t}$ and $\widehat{var}(\widehat{R_t})$ via Equation (3.5, 3.6, 3.7, 3.8, 3.9);
    `// Stopping`
    if $\widehat{R_t}$ and $\widehat{var}(\widehat{R_t})$ satisfy stopping strategy **then**
        stop = True;
    end if
end while

list, and y denote the relevance label; then we can construct a distribution of random variable (y_i, y_j) over position pairs (i, j) that satisfies the expectation $\mathbb{E}(y_i, y_j)$ equals AP; finally we add up all the probabilities associated with all pairs involving a given position r and get $p(r) = \frac{1}{2N}(1 + \frac{1}{r} + \frac{1}{r+1} + \ldots + \frac{1}{N}) \approx \frac{1}{2N} \log \frac{N}{r}$. For more details, the readers are referred to [13, 126].

Thus the AP-prior distribution is defined as follows:

$$p_i = \frac{1}{Z} \log \frac{N}{r_i} \quad r_i \in [1, N]\,, \tag{3.1}$$

where $Z = \sum_{i=1}^{N} \log \frac{N}{r_i}$ is the normalization factor.

Inclusion probability. In order to have a simple expression for *inclusion probability*, we adopt *sampling with replacement* as our sampling method. At each iteration t and for each draw, a document is sampled independently from one of the aforementioned distributions. Let *selection probability* denote the probability that a document is sampled for a draw, and *inclusion probability* the probability that a document is in-

cluded in the sample set considering all the draws. The first-order and second-order inclusion probabilities under sampling with replacement are indispensable to calculate $\widehat{R}$. Under sampling-with-replacement design, the first-order inclusion probability π_i is given by:

$$\pi_i = 1 - \prod_{t=1}^{T} \left(1 - p_i^t\right)^{n_t} . \tag{3.2}$$

The second-order inclusion probability $\pi_{i,j}$ – the probability of any two different document d_i and d_j being included – is given by:

$$\pi_{i,j} = \pi_i + \pi_j - \left[1 - \prod_{t=1}^{T} \left(1 - p_i^t - p_j^t\right)^{n_t}\right] . \tag{3.3}$$

3.3.3 Estimate R and $var(\widehat{R})$

As mentioned in Section 3.1, one of our goals is to estimate R – the total number of relevant documents in the document collection for a given topic. In this section, we first clarify the exact expression of R with regard to the population, then introduce its estimators on the sample set. Also for notation simplicity sometimes we omit the subscript t.

Let $\mathcal{C}_q = \sum_{i=1}^{N} d_i$ denote a population of documents and let y_i be an indicator variable of d_i , with $y_i = 1$ if the document d_i is relevant, and $y_i = 0$ otherwise. The population total is defined as the summation of all y_i , i.e., the total number of relevant documents in the collection. We write R as follows:

$$R = \sum_{i=1}^{n} y_i . \tag{3.4}$$

Suppose our framework produces a sample set at each iteration t, denoted by S_t. Let $\widetilde{S}_t$ denote the cumulated sample set till iteration t, $\widetilde{S}_t = \cup_{k=1}^{t} S_k$. Furthermore, we also let $\widetilde{S}'_t$ denote the subset of $\widetilde{S}_t$ and $\widetilde{S}'_t$ only contains unique documents. Correspondingly, let n_t be the number of documents in S_t, $\widetilde{n}_t$ be the number of documents in $\widetilde{S}_t$, $\widetilde{n}'_t$ be the number of documents in $\widetilde{S}'_t$.

Note that when sampling documents, each document has different sampling probability compared to other ones, therefore we can not apply a simple estimator for equal sampling probability (e.g., $\frac{N}{\widetilde{n}_t} \sum_{i \in \widetilde{S}_t} y_i$) to estimate R. Instead we employ Horvitz-Thompson estimator and Hansen-Hurwitz estimator, both of which can estimate R as well as $var(\widehat{R})$.

Both of them are designed for sampling with unequal probabilities, Hansen-Hurwitz estimator is only restricted for with-replacement sampling, while Horvitz-Thompson estimator is for any design. On the other hand, the inclusion probability of Horvitz-Thompson estimator is not easy to calculate when the sampling design is complex. One can choose to use one of the estimators according to their sampling design. We provide the proof of unbiasedness of the two estimators under our sampling design in Section 3.3.4. We also provide the theoretical conditions of zero variance estimators in this section. For more details of the derivation the reader can refer to [158].

Horvitz-Thompson Estimator

The Horvitz-Thompson estimator provides an unbiased estimator of population total under any sampling design, with or without replacement [72]. It is written as $\widehat{\tau} = \sum_{i \in S'} \frac{y_i}{\pi_i}$, where S' is a subset of S and S' only contains unique units, and π_i is the inclusion probability for unit i.

In our case, the population total is R. The Horvitz-Thompson estimator of R on $\widetilde{S}'_t$ is written as:

$$\widehat{R}_t^{HT} = \sum_{i \in \widetilde{S}'_t} \frac{y_i}{\pi_i} \,. \tag{3.5}$$

Furthermore, it is good to know whether the estimator is stable. An estimator with low variance is desirable. In order to calculate the variance we need to conduct the same sampling procedure many times, which is not feasible in practice. Luckily, there is an unbiased estimator of the variance of the population total estimator, given by:

$$\widehat{var1}\left(\widehat{R}_t^{HT}\right) = \sum_{i \in \widetilde{S}'_t} \left(\frac{1}{{\pi_i}^2} - \frac{1}{\pi_i}\right) {y_i}^2 + 2 \sum_{i > j \in \widetilde{S}'_t} \left(\frac{1}{\pi_i \pi_j} - \frac{1}{\pi_{ij}}\right) y_i y_j \,. \tag{3.6}$$

As $\widehat{var1}(\widehat{R}_t^{HT})$ may produce negative values, whereas the true variance must be non-negative. Hence, we also adopt another estimator of variance,[2] given by:

$$\widehat{var2}\left(\widehat{R}_t^{HT}\right) = \frac{N - \widetilde{n}'_t}{N \widetilde{n}'_t} \frac{1}{\widetilde{n}'_t - 1} \sum_{i \in \widetilde{S}'_t} \left(\widetilde{n}'_t \frac{y_i}{\pi_i} - \widehat{R}_t^{HT}\right)^2 \,. \tag{3.7}$$

Note that when $\pi_i = \frac{\widetilde{n}'_t y_i}{N}$, the variance of $\widehat{R}_t^{HT}$ equals 0 [72].

Hansen-Hurwitz Estimator

Hansen-Hurwitz estimator provides an unbiased estimator of population total under sampling with replacement, and the sampling distribution should be the same for each draw [158]. In our case, the sampling distribution changes at each iteration and converges to the ultimate distribution produced by the ranking model trained on the whole documents. This is the major difference with the setting of the vanilla Hansen-Hurwitz estimator. In Section 3.3.4 we will prove that under varying sampling distributions at different iterations, $\widehat{R}_t^{HH}$ is also unbiased.

The Hansen-Hurwitz estimator of R on $\widetilde{S}_t$ is expressed as follows[3]:

$$\widehat{R}_t^{HH} = \frac{1}{\widetilde{n}_t} \sum_{k=1}^{t} \sum_{i \in S_k} \frac{y_i}{p_i^k} \,. \tag{3.8}$$

[2] It is biased yet "conservative" (meaning it tends to be larger than the actual variance). The intuition is taking the Horvitz-Thompson estimator as the average of i.i.d. random variable .

[3] Note that if taken in the view of *importance sampling*, Hansen-Hurwitz estimator equals to the estimation of population total obtained by importance sampling, denoted by $q_i = \widetilde{n}'_t \frac{y_i}{\pi_i}$. The sample variance of q_i is defined by $s^2 = \frac{1}{\widetilde{n}'_t - 1} \sum_{i=1}^{\widetilde{n}_t} (q_i - \widehat{R}_t^{HT})^2$. The alternative variance estimator is $\widehat{var}(\widehat{R}_t^{HT}) = \frac{N - \widetilde{n}_t}{N \widetilde{n}'_t} s^2$ [158].

Similarly, there is also an estimator of the variance of the population total estimator, given by:

$$\widehat{var}\left(\widehat{R_t}^{HH}\right) = \frac{1}{\widetilde{n}_t\left(\widetilde{n}_t - 1\right)} \sum_{k=1}^{t} \sum_{i \in S_k} \left(\frac{y_i}{p_i^k} - \widehat{R_t}^{HH}\right)^2 . \tag{3.9}$$

Note that when $\frac{y_i}{p_i^k}$ is a constant, the variance of the Hansen-Hurwitz estimator equals 0 [158, page 68].

For both Horvitz-Thompson and Hansen-Hurwitz, there is no guarantee that their estimators of variance are unbiased.

3.3.4 Unbiasedness

Hansen-Hurwitz Estimator

Lemma 1. *Given that the sampling design is with replacement, and at each draw l, a unit is independently sampled from a different distribution $\{p_i^l\}$ from the previous draws. Let S denote the sample set. The Hansen-Hurwitz estimator of population total is unbiased, i.e., $\mathbb{E}\left(\sum_{y_i^l \in S} \frac{y_i^l}{p_i^l}\right) = \tau$.*

Proof. First consider the one unit case. Assume there is only one unit in S, thus there are N possible sample set for S. Let y_S denote the sample value and p_S the selection probability, thus Hansen-Hurwitz estimator can be rewritten as $\tau_S = \frac{y_S}{p_S}$. Its expected value over all possible sample sets is:

$$\mathbb{E}(\tau_S) = \sum_{S=1}^{N} \tau_S p_S = \sum_{S=1}^{N} y_S = \tau . \tag{3.10}$$

For the n unit case, take each unit i as a sample set with only one unit, which is denoted by S_i. Thus $\mathbb{E}(\tau_{S_i}) = \tau$.

As each S_i is independent from any other S_j, we have

$$\begin{aligned} \mathbb{E}(\tau_S) &= \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} \tau_{S_i}\right) && (3.11)\\ &= \frac{1}{n}\left(\sum_{i=1}^{n} \mathbb{E}\left(\tau_{S_i}\right)\right) && (3.12)\\ &= \frac{n\tau}{n} = \tau . && (3.13) \end{aligned}$$

Equation (3.11) to Equation (3.12) is because $\{\tau_{S_i}\}$ is a set of independent variables, thus we can swap expectation and summation. Equation (3.12) to Equation (3.13) is because Equation (3.10) holds. □

Horvitz-Thompson Estimator

The unbiasedness of Horvitz-Thompson estimator is proved [158, page 76]. We rephrase the proof below for reader's convenience.

Lemma 2. *Given that the sampling design is with replacement, and at each iteration t, a set of units are independently sampled from a different distribution $\{p_i^t\}$ from the previous iterations. Let S' denote the sample set where all the duplicated units are removed. The Horvitz-Thompson estimator of population total is unbiased, i.e.,* $\mathbb{E}\left(\sum_{y_i \in S'} \frac{y_i}{\pi_i}\right) = \tau$.

Proof. Let random variable z_i denote whether the i-th unit is included in the sample set: $z_i = 1$ means that the i-th unit is in the sample set and $z_i = 0$ means that the i-th unit is not in the sample set. For any unit i, z_i is a Bernoulli random variable, thus $\mathbb{E}z_i = p(z_i = 1) = \pi_i$.

The Horvitz-Thompson estimator can be rewritten as follows, where each y_i is regarded as a constant:

$$\sum_{y_i \in S'} \frac{y_i}{\pi_i} = \sum_{i=1}^{N} \frac{y_i z_i}{\pi_i} \,. \tag{3.14}$$

The expected value is:

$$\begin{aligned} \mathbb{E}\left(\sum_{y_i \in S'} \frac{y_i}{\pi_i}\right) &= \mathbb{E}\left(\sum_{i=1}^{N} \frac{y_i z_i}{\pi_i}\right) && (3.15)\\ &= \sum_{i=1}^{N} \frac{y_i \mathbb{E}z_i}{\pi_i} && (3.16)\\ &= \sum_{i=1}^{N} \frac{y_i \pi_i}{\pi_i} = \sum_{i=1}^{N} y_i = \tau \,. && (3.17) \end{aligned}$$

Equation (3.15) to Equation (3.16) is because $\{\frac{y_i z_i}{\pi_i}\}$ is a set of independent variables, thus we can swap expectation and summation. Equation (3.16) to Equation (3.17) is because $\mathbb{E}z_i = p(z_i = 1) = \pi_i$. □

3.3.5 Stopping Strategy

We propose two stopping strategies based on $\widehat{R}$ and its estimated variance, $\widehat{var}(\widehat{R})$. They represent different levels of conservativeness in terms of deciding when to stop the TAR process.

For notational simplicity, we omit the subscript t. Let r denote the number of unique relevant documents that have already been read by the reviewer. Let $\widehat{R}$ denote the estimated total number of relevant documents, $\widehat{var}(\widehat{R})$ the estimated variance, and $recall_t$ the target recall specified by the reviewer.

Optimistic This strategy examines whether $\frac{r}{recall_t} \geq \widehat{R}$, and if so stops the TAR process. The intuition is straightforward: if we have collected more relevant documents than the target number we estimated, we should feel confident to stop.

Conservative This strategy examines whether $\frac{r}{recall_t} \geq \widehat{R} + \sqrt{\widehat{var}(\widehat{R})}$, and if so stops the TAR process. This inequality is more conservative than the first one, that is, it

continues providing document to the reviewer until it accumulates a higher confidence that the target recall is reached.

3.4 Three New Benchmark Collections

In this section, we introduce three new benchmark collections that we contributed to the research community. We first talk about the design goal of the three benchmark collections; then we introduce the criterion to select topics, the method to extract documents for the document collection, and the resource from which we extract relevance labels. Finally, we show general statistics of the three benchmark collections.

3.4.1 Retrieval Approaches for Conducting Systematic Reviews

Evidence-based medicine has become an important pillar in health care and policy making [116]. In order to practice evidence-based medicine, it is important to have a clear overview over the current scientific consensus. These overviews are provided in systematic review articles that summarize all available evidence regarding a certain topic (e.g., a treatment or diagnostic test). In order to write a systematic review, researchers have to conduct a search that will retrieve all the studies that are relevant. The large and growing number of published studies, and their increasing rate of publication, makes the task of identifying relevant studies in an unbiased way both complex and time consuming to the extent that jeopardizes the validity of their findings and the ability to inform policy and practice in a timely manner. Hence, the need for automation in this process becomes of utmost importance. Finding all relevant studies in a corpus is a difficult task, known in the information retrieval domain as the *total recall* problem [60, 135], which we have introduced in Chapter 1.

To this date, retrieval of evidence to inform systematic reviews is being conducted in multiple stages. The first stage is *Boolean search*. Information specialists (or experts) build a broad Boolean query expressing what constitutes relevant information. The query is then submitted to a medical database containing titles, abstracts, and indexing terms of a controlled vocabulary of medical studies. The result is a set of potentially interesting studies. The second stage is *title and abstract screening*. Experts screen the titles and abstracts of the returned set and decide which one of those hold potential value for their systematic review. The third stage is *study screening*. Experts download the full text of the potentially relevant abstracts, identified in the previous phase and examine the content to decide whether indeed these studies are relevant or not. The result of the second screening is a set of references to be included in the systematic review.

Unfortunately, the precision of the Boolean searches is typically low, hence reviewers often need to look manually through many thousands of irrelevant titles and abstracts in order to identify a small number of relevant ones. Furthermore, the recall of the searches is often assumed to be 100%, which may not be the case. To overcome some of the limitations of the Boolean search, researchers have been testing the effectiveness of machine learning and information retrieval methods [116]. O'Mara-Eves et al. [124] provide a systematic review of the use of text mining techniques for study identification in systematic reviews.

3.4.2 Goals of Benchmark Collections

The construction of the three benchmark collections took three years. They were released with the CLEF Technology-Assisted Reviews in Empirical Medicine tracks in 2017, 2018 and 2019 [81, 83, 85]. The tracks focused on retrieving studies for conducting systematic reviews. Retrieval in this area is generally considered very difficult, where sensitive searches result in large quantities of references to be screened manually, and a breakthrough in this field would likely be applicable to other areas as well. The task has a focus on the second stage of the process, i.e., given the results of a Boolean search how to make abstract and title screening more effective and efficient. Currently, a typical number needed to read, the number of studies to screen to identify 1 eligible study, for systematic reviews is approximately 80 when applied to potential abstracts that need further full text assessment. With an average of 7000 results to be screened, which would take approximately 120 hours to screen (1 minute per abstract [145]), a huge benefit can be made in reducing the workload in this process.

The ultimate goal of the tracks is to provide an experimental platform for automatic methods to retrieve relevant studies with high precision and high recall, and release a reusable test collection that can be used as a reference for comparing different retrieval and mining approaches in the field of medical systematic reviews. Specifically, given the results of the Boolean search from the first stage as the starting point, a retrieval system is expected to rank the set of the provided abstracts. The task has two goals: (1) to produce an efficient ordering of the documents, such that all the relevant abstracts are retrieved above the irrelevant ones, and (2) to identify the relevant subset of abstracts to be shown to a user, that is a stopping point in the ranked list of abstract, where a researcher could confidently stop screening abstracts and titles.

3.4.3 Documents, Qrels, and Topics

The tasks defined in the three tracks are slightly evolving, but the released test collections follow the same Cranfield paradigm. In this section, we omit the small differences of the three test collections, we only introduce common points including the criterion to select topics, the method to extract documents for the document collection, and the resource from which we extract relevance labels.

Documents

The document collection is PubMed Baseline Repository, available under the website of the National Center for Biotechnology Information.[4] PubMed comprises more than 30 million citations for biomedical literature from MEDLINE, life science journals, and online books. A set of MEDLINE or PubMed citation records in XML format is provided for downloading on an annual basis. The annual baseline is released in December of each year.

Qrels

For the construction of the *qrel* files, we considered the reference section of the systematic review articles. The references in the articles are split into three categories:

[4] `ftp://ftp.ncbi.nlm.nih.gov/pubmed/baseline`

included, exclude, and additional. Included are the studies that are relevant to the systematic review. Excluded are the studies that in the abstract and title screening stage were considered relevant, but at the article screening phase were considered irrelevant to the study and hence excluded from it. Additional are additional references that do not impact the outcome of the study, and hence are irrelevant to it. The included references were the relevant studies at the document-level qrels, while both the included and excluded references were considered relevant at the abstract-level qrels. The format of the qrels followed the standard four columns in the TREC format: *topic*, *iteration*, *document*, and *relevance*. The *topic* is the topic ID of the systematic review, the *iteration* in our case is a dummy field always zero and not used, the *document* is the PMID, and the *relevance* is a binary code of 0 for not relevant and 1 for relevant studies. The order of documents in the qrel files is not indicative of relevance. Studies that were returned by the Boolean query but were not relevant based on the above process, were considered irrelevant. Those are studies that were excluded at the abstract and title screening phase. All other documents in MEDLINE were also assumed to be irrelevant, given that they were not judged by the human assessor.

Topics

To select the topics in the benchmark collections, we considered four types of systematic reviews in total. They are diagnostic test accuracy, intervention, prognosis, and qualitative systematic reviews. These reviews are publicly available through the Cochrane library.[5]

Topic file defines the information need of a systematic review. A topic file is generated through the following procedure. For each systematic review, we reviewed the search strategy from the corresponding study in the Cochrane library. A search strategy, among others, consists of the exact Boolean query developed and submitted to a medical database, at the time the review was conducted, and typically can be found in the Appendix of the study. Rene Spijker, a co-organizer of the track and a Cochrane information specialist examined the grammatical correctness of the search query and specified the date range which dictated the valid dates for the articles to be included in this systematic review. The date range was necessary because a study published after the systematic review should not be included even though it might be relevant, since that would require manually examining its content to quantify its relevance. Note that a number of medical databases, and search interfaces to these databases is available for search, and for each one information specialists construct a different variation of their query that better fits the data and meta-data of the database. For this task, we only considered the Boolean query constructed for the MEDLINE database, using the Wolters Kluwer Ovid interface.[6] Then we submitted the constructed Boolean query to the interface and collected all the returned PubMed document identification numbers (PMIDs) which satisfied the date range constraint. This step was automated by a Python script we put together and through an interface available to the University of Amsterdam.[7]

[5] `http://www.cochranelibrary.com/`
[6] `http://demo.ovid.com/demo/ovidsptools/launcher.htm`
[7] `https://github.com/dli1/tar_data_collection`

A topic file is in a text format and contains four sections, *topic*, *title*, *query*, and *PMIDs*, where the *topic* is the topic ID, which is a substring of DOI of the systematic review (e.g., CD010438 for 10.1002/14651858.CD010438.pub2), *title* is the title of the systematic review, *query* is the Boolean query included in the systematic review, and *PMIDs* are the document IDs returned by the Boolean query. The PMIDs can be used to access the corresponding document text through the website of the National Center for Biotechnology Information.[8] An example of a topic file can be viewed below.

```
Topic: CD009551
Title: Polymerase chain reaction blood tests for the diagnosis
 of invasive aspergillosis in immunocompromised people

Query:
exp Aspergillosis/
exp Pulmonary Aspergillosis/
exp Aspergillus/
(aspergillosis or aspergillus or aspergilloma or "A.fumigatus" or
"A. flavus" or "A. clavatus" or "A. terreus" or "A. niger").ti,ab.
or/1-4
exp Nucleic Acid Amplification Techniques/
pcr.ti,ab.
"polymerase chain reaction*".ti,ab.
or/6-8
5 and 9
exp Animals/ not Humans/
10 not 11

Pids:
    25815649
    26065322
    ...
```

3.4.4 Statistics of Benchmark Collections

The Test Collection in the 2017 CLEF Technology-Assisted Reviews in Empirical Medicine Track

We considered 58 systematic reviews on diagnostic test accuracy, which were all the available systematic reviews when the track was organized [81]. Out of the 58 reviews 8 were discarded since the provided Boolean query was not in the right format, which made it difficult if not impossible to reconstruct the set of PMIDs, hence the 50 topics, of which 20 topics were randomly selected in the development and the remaining 30 were selected in the test set.

Table 3.2 shows the distribution of the relevant documents at the abstract or document level for all the topics in the development set and the test set. The total number of unique PMID is 149,405 for the development set and 117,562 for the test set. Their percentages of relevant documents at the abstract level are quite close, which is 1.88% for the development set and 1.58% for the test set. This is not true at the document level, however, where the fraction of relevant documents in the test set is almost twice as large as in the development set, even though there are 0.52% and 0.33% of relevant

[8] https://www.ncbi.nlm.nih.gov/books/NBK25497/

studies, respectively. In [141], a test collection was developed based on a random selection of 93 Cochrane systematic reviews (not just diagnostic test accuracy reviews), and reported a slightly higher rate of relevance ($\frac{14}{1159} = 1.2\%$). However, compared with the TREC campaign, the rate of relevant documents is 5.45% for the ad-hoc track of TREC 8 and 2.78% for the web track of TREC 2002. Overall, the number of relevant documents is not very high, making locating them quite a difficult task.

As one can observe in Table 3.2, there are topics for which the output of the Boolean query is rather narrow, with as few as 64 studies to be reviewed for topic CD008760. Cochrane is conducting systematic reviews on a regular basis, in an attempt to update each review every two-three years. Some of the reviews considered for the construction of the benchmark collection, such as the CD008760 review, are updates to previous reviews. These updates, only specify a query for a time range that starts after the last review on the topic was conducted. Hence, the 64 studies, are the output of the Boolean query for this short time range, hence its small number. If the Boolean query were to run against the entire MEDLINE database, the number of studies would be in the range of tens of thousands, as is the case for some other reviews considered, e.g., CD008782.

The Test Collection in the 2018 CLEF Technology-Assisted Reviews in Empirical Medicine Track

At the time of the topic construction of this track, 88 systematic reviews were available; 50 of them were used in the 2017 task [81], and out of the remaining 38, 30 were chosen to constitute the 2018 topic set [83]. The track provided two sets of topics: (1) a development set, and (2) a test set. The development set consisted of 42 topics out of the 50 topics provided in the 2017 version of the lab. The 50 topics released in 2017 were re-examined by a Cochrane information specialist, Rene Spijker, and 8 of them were found not suitable for training or testing purposes, and hence removed from the development set. The IDs of the 8 topics are the following: CD007431, CD010772, CD010775, CD010896, CD010771, CD011145, CD010783, and CD010860, where in parenthesis is the filename of the topic in the 2017 release of the data.

The development and test systematic reviews can be found in Tables 3.3 and 3.4. The tables provide the topic id, a substring of DOI of the document (e.g., the DOI for topic ID CD008122 is `10.1002/14651858.CD008122.pub2`), the title of the systematic review that corresponds to the topic, and the publication date.

The Test Collection in the 2019 CLEF Technology-Assisted Reviews in Empirical Medicine Track

Since search and classification algorithms were developed demonstrating good retrieval performance over the diagnostic test accuracy studies since the 2017 track, the 2019 track extended its focus to intervention, prognosis, and qualitative systematic reviews [85].

To construct the benchmark collection, we used 8 diagnostic test accuracy, 20 intervention, 1 prognosis, and 2 qualitative systematic reviews already conducted by Cochrane researchers. These reviews can be found in the Cochrane library. 72 diagnostic

Table 3.2: Statistics of topics in the development and test set in the 2017 CLEF Technology-Assisted Reviews in Empirical Medicine Track.

Topic	# total PMIDs	# abs rel	# doc rel	% abs rel	% doc rel
Development Set					
CD010438	3250	39	3	1.20	0.09
CD007427	1521	123	17	8.09	1.12
CD009593	14922	78	24	0.52	0.16
CD011549	12705	2	1	0.02	0.01
CD011134	1953	215	49	11.01	2.51
CD008686	3966	7	5	0.18	0.13
CD011975	8201	619	60	7.55	0.73
CD009323	3881	122	9	3.14	0.23
CD009020	1584	162	12	10.23	0.76
CD011548	12708	113	5	0.89	0.04
CD011984	8192	454	28	5.54	0.34
CD010409	43363	76	41	0.18	0.09
CD008054	3217	274	41	8.52	1.27
CD010771	322	48	1	14.91	0.31
CD009591	7991	144	41	1.80	0.51
CD008691	1316	73	20	5.55	1.52
CD010632	1504	32	14	2.13	0.93
CD007394	2545	95	47	3.73	1.85
CD008643	15083	11	4	0.07	0.03
CD009944	1181	117	64	9.91	5.42
Test Set					
CD007431	2074	24	15	1.16	0.72
CD008803	5220	99	99	1.90	1.90
CD008782	10507	45	34	0.43	0.32
CD009647	2785	56	17	2.01	0.61
CD009135	791	77	19	9.73	2.40
CD008760	64	12	9	18.75	14.06
CD010775	241	11	4	4.56	1.66
CD009519	5971	104	46	1.74	0.77
CD009372	2248	25	10	1.11	0.44
CD010276	5495	54	24	0.98	0.44
CD009551	1911	46	16	2.41	0.84
CD012019	10317	3	1	0.03	0.01
CD008081	970	26	10	2.68	1.03
CD009185	1615	92	23	5.70	1.42
CD010339	12807	114	9	0.89	0.07
CD010653	8002	45	0	0.56	0.00
CD010542	348	20	8	5.75	2.30
CD010896	169	6	3	3.55	1.78
CD010023	981	52	14	5.30	1.43
CD010772	316	47	11	14.87	3.48
CD011145	10872	202	48	1.86	0.44
CD010705	114	23	18	20.18	15.79
CD010633	1573	4	3	0.25	0.19
CD010173	5495	23	10	0.42	0.18
CD009786	2065	10	6	0.48	0.29
CD010386	626	2	1	0.32	0.16
CD010783	10905	30	11	0.28	0.10
CD010860	94	7	4	7.45	4.26
CD009579	6455	138	79	2.14	1.22
CD009925	6531	460	55	7.04	0.84

Table 3.3: Statistics of topics in the development set in the 2018 CLEF Technology-Assisted Reviews in Empirical Medicine Track.

Topic	# total PMIDs	# abs rel	# doc rel	% abs rel	% doc rel
Development Set					
CD010438	3250	39	3	1.20	0.09
CD007427	1521	123	17	8.09	1.12
CD009593	14922	78	24	0.52	0.16
CD011549	12705	2	1	0.02	0.01
CD011134	1953	215	49	11.01	2.51
CD008686	3966	7	5	0.18	0.13
CD011975	8201	619	60	7.55	0.73
CD009323	3881	122	9	3.14	0.23
CD009020	1584	162	12	10.23	0.76
CD011548	12708	113	5	0.89	0.04
CD011984	8192	454	28	5.54	0.34
CD010409	43363	76	41	0.18	0.09
CD008054	3217	274	41	8.52	1.27
CD009591	7991	144	41	1.80	0.51
CD008691	1316	73	20	5.55	1.52
CD010632	1504	32	14	2.13	0.93
CD007394	2545	95	47	3.73	1.85
CD008643	15083	11	4	0.07	0.03
CD009944	1181	117	64	9.91	5.42
CD008803	5220	99	99	1.90	1.90
CD008782	10507	45	34	0.43	0.32
CD009647	2785	56	17	2.01	0.61
CD009135	791	77	19	9.73	2.40
CD008760	64	12	9	18.75	14.06
CD009519	5971	104	46	1.74	0.77
CD009372	2248	25	10	1.11	0.44
CD010276	5495	54	24	0.98	0.44
CD009551	1911	46	16	2.41	0.84
CD012019	10317	3	1	0.03	0.01
CD008081	970	26	10	2.68	1.03
CD009185	1615	92	23	5.70	1.42
CD010339	12807	114	9	0.89	0.07
CD010653	8002	45	0	0.56	0.00
CD010542	348	20	8	5.75	2.30
CD010023	981	52	14	5.30	1.43
CD010705	114	23	18	20.18	15.79
CD010633	1573	4	3	0.25	0.19
CD010173	5495	23	10	0.42	0.18
CD009786	2065	10	6	0.48	0.29
CD010386	626	2	1	0.32	0.16
CD009579	6455	138	79	2.14	1.22
CD009925	6531	460	55	7.04	0.84

Table 3.4: Statistics of topics in the test set in the 2018 CLEF Technology-Assisted Reviews in Empirical Medicine Track.

Topic	# total PMIDs	# abs rel	# doc rel	% abs rel	% doc rel
Test Set					
CD008122	1911	272	57	0.142	0.030
CD012599	8048	575	19	0.071	0.002
CD009175	5644	65	7	0.012	0.001
CD009694	161	16	9	0.099	0.056
CD009263	79786	124	10	0.002	0.000
CD010502	2985	229	71	0.077	0.024
CD010680	8405	26	0	0.003	0.000
CD010864	2505	44	3	0.018	0.001
CD011431	1182	297	26	0.251	0.022
CD011602	6157	8	1	0.001	0.000
CD011420	251	42	5	0.167	0.020
CD011686	9443	55	2	0.006	0.000
CD012179	9832	304	117	0.031	0.012
CD012281	9876	23	9	0.002	0.001
CD011053	2235	12	7	0.005	0.003
CD011515	7244	127	1	0.018	0.000
CD008587	9158	79	35	0.009	0.004
CD011926	4050	40	29	0.010	0.007
CD012165	10222	308	47	0.030	0.005
CD012083	322	11	5	0.034	0.016
CD008892	1499	69	30	0.046	0.020
CD011126	6000	13	9	0.002	0.002
CD010657	1859	139	35	0.075	0.019
CD008759	932	60	42	0.064	0.045
CD010296	4602	53	38	0.012	0.008
CD010213	15198	599	33	0.039	0.002
CD012009	536	37	4	0.069	0.007
CD011912	1406	36	18	0.026	0.013
CD012010	6830	290	8	0.042	0.001
CD012216	217	11	1	0.051	0.005

test accuracy reviews used in the 2017 and 2018 versions of the lab, as well as 20 different Intervention reviews were also collected and made available to the participants as a development set.

The average percentage of relevant abstract in the development set is 6.5% of the total number of PMIDs released, and in the test set 8.9%, while at the content level the average percentage is 2.6% in the development set, and 3.9% in the test set. Table 3.5 shows the distribution of the relevant documents at abstract and document level for all the topics in the test set. A break down of the average percentage of relevant abstracts/documents are: diagnostic test accuracy 12.9%/5.3%, intervention 7.6%/3.4%, prognosis 15.7%/9.4%, qualitative 2.6%/1.0%.

Table 3.5: Statistics of topics in the test set in the 2019 CLEF Technology-Assisted Reviews in Empirical Medicine Track.

Topic	# total PMIDs	# abs rel	# doc rel	% abs rel	% doc rel
Diagnostic Test Accuracy					
CD008874	2382	130	121	0.055	0.051
CD009044	3169	47	8	0.015	0.003
CD011686	9729	74	3	0.008	0.000
CD012080	6643	85	85	0.013	0.013
CD012233	472	54	10	0.114	0.021
CD012567	6735	12	5	0.002	0.001
CD012669	1260	82	31	0.065	0.025
CD012768	131	100	41	0.763	0.313
Intervention					
CD000996	281	10	6	0.036	0.021
CD001261	571	85	26	0.149	0.046
CD004414	336	32	13	0.095	0.039
CD006468	3874	91	15	0.023	0.004
CD007867	943	31	15	0.033	0.016
CD009069	1757	94	6	0.054	0.003
CD009642	1922	90	72	0.047	0.037
CD010038	8867	36	12	0.004	0.001
CD010239	224	23	12	0.103	0.054
CD010558	2815	75	16	0.027	0.006
CD010753	2539	35	21	0.014	0.008
CD011140	289	4	0	0.014	0.000
CD011571	146	21	6	0.144	0.041
CD011768	9160	81	31	0.009	0.003
CD011977	195	65	38	0.333	0.195
CD012069	3479	425	327	0.122	0.094
CD012164	61	10	3	0.164	0.049
CD012342	2353	9	0	0.004	0.000
CD012455	1593	12	5	0.008	0.003
CD012551	591	86	34	0.146	0.058
Prognosis					
CD012661	3367	527	317	0.157	0.094
Qualitative					
CD011558	2168	51	27	0.024	0.012
CD011787	4369	125	34	0.029	0.008

3.5 Experimental Setup

3.5.1 Research Questions

The goal of the TAR process is to identify as many relevant documents as possible with minimal cost of returned irrelevant documents, and to stop exactly at the target recall specified by the reviewer. Based on this, our research questions are as follows.
RQ1 Can the framework *autostop* proposed in Section 3.3 achieve target recall with minimal assessment cost and stop the document selection process on time?
RQ2 Whether the estimated $\widehat{R}$ using the Horvitz-Thompson estimator (Section 3.3.3) and the Hansen-Hurwitz estimator (Section 3.3.3) are unbiased with low variance?
RQ3 How does the proposed framework trade off high recall against accurate $\widehat{R}$?
RQ4 How does the estimation module (Section 3.3.3) and the stopping module (Section 3.3.5) impact the performance of the model?

3.5.2 Dataset

We test our framework on a wide range of datasets including the CLEF Technology-Assisted Reviews in Empirical Medicine datasets (EMED for short), the TREC Total Recall datasets (TR for short), and the TREC Legal datasets (LEGAL for short).

EMED dataset. The Technology-Assisted Reviews in Empirical Medicine Task at CLEF aims at evaluating search algorithms that seek to identify all studies relevant for conducting a systematic review in empirical medicine.

Three datasets have been released namely EMED 2017 [81], EMED 2018 [83, 154] and EMED 2019 [85]. EMED 2017 and EMED 2018 only include Diagnostic Test Accuracy reviews, while EMED 2019 includes three new review types, which are Intervention, Prognosis, and Qualitative systematic reviews. We use all the Diagnostic Test Accuracy reviews in the three datasets for our experiments. There are 42, 30 and 31 reviews (topics), respectively. We select 12 topics in EMED 2017 as the training topics for hyper-parameter fine-tuning of baselines and our method. The 12 topics are selected in the way that their *prevalence* values ranging from small to large value. We use the 30 topics in EMED 2017, 30 topics in EMED 2018 and 31 topics in EMED 2019 as the test topics.

For each topic, a topic description, a subset of the PubMed abstracts which are related to the topic and need to be ranked, along with the relevance judgments of the studies in this set are provided. Each topic description provides a topic title, which is the title of the corresponding systematic review article, like “Galactomannan detection for invasive aspergillosis in immunocompromised patients." The titles usually contain terms in biomedical literature, not like general queries users submit to search engine in their everyday life. The document collection – PubMed baseline repository[9] – comprises biomedical literature from MEDLINE, life science journals, and online books.

TR dataset. The TREC Total Recall track [60] aims to evaluate methods designed to achieve very high recall – as close as 100% – with a human assessor in the loop.

We use the *athome1* and *athome4* dataset provided in the TREC 2016 Total Recall track. *Athome1* contains 10 topics and *athome4* contains 34 topics[10] We use the 10

[9] `ftp://ftp.ncbi.nlm.nih.gov/pubmed/baseline`
[10] The TR dataset is publicly available under `https://plg.uwaterloo.ca/~gvcormac/total-`

topics in *athome1* as the training topics and the 34 topics in *athome4* as the test topics.

Each topic supplies a short topic title, typically one to three words, like "Olympics", "bottled water". The topics are similar to queries people submit to search engines in their everyday life. The document collection consists of the Jeb Bush's emails. It consists of 290,099 emails from Jeb Bush's eight-year tenure as Governor of Florida.

LEGAL dataset. The TREC Legal track [41] focuses on evaluation of search technology for discovery of electronically stored information in litigation and regulatory settings. We use the dataset provided in the interactive task in the TREC 2010 Legal track.[11] It includes topic 301, 302, 303, 304. As we did not find any other topics that have the same document collection and relevancy labels, we use 301 and 302 as the training topics and 303 and 304 as the test topics.

Each topic consists of a mock complaint and document requests, where the mock complaint sets forth the legal and factual basis for the hypothetical lawsuit that motivates the discovery requests, and the document requests specify the categories of documents that must be located and produced. The document collection is a processed variant of the Enron email dataset, containing about 685,592 email messages captured by the federal energy review commission from Enron, in the course of its investigation of Enron's collapse. Similarly with the TR dataset, we use the same strategy to generate the subset of documents for each topic.

Table 3.6, 3.7 and 3.8 list the statistics of the datasets used in our experiments. For more details of topic splits readers can refer to Table 3.9. We calculated *prevalence*, which is defined as the percentage of relevant documents in the given documents for a topic. Overall the prevalence is low for all the datasets, ranging from 0.10 % to 2.15%, indicating it is not an easy task to retrieve all the relevant documents. Figure 3.2 further shows the box plot of topic prevalence of the 5 test datasets. The three EMED datasets have larger prevalence than TR, followed by LEGAL.

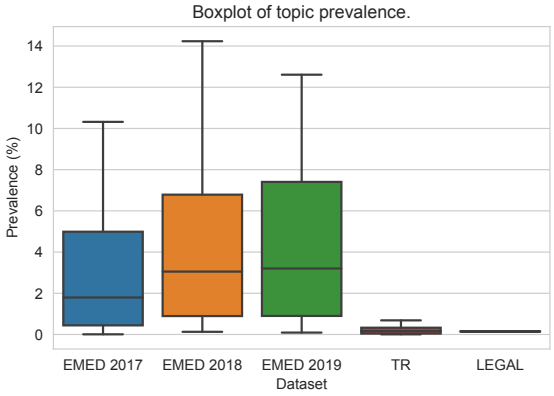


Figure 3.2: Box plot of topic prevalence of the 5 datasets.

recall/, conditioned on a usage agreement.

[11] https://trec-legal.umiacs.umd.edu

Table 3.6: EMED dataset statistics

	train	test (EMED 2017)	test (EMED 2018)	test (EMED 2019)
# doc per topic	4494.17	6257.97	7292.73	2658.74
# topic	12	30	30	31
avg doc length	190.37	182.26	205.41	236.92
# rel per topic	95.67	94.47	132.43	54.26
% rel per topic	0.97	1.80	2.05	2.15

Table 3.7: TR dataset statistics

	train	test
# doc per topic	290099	290099
# topic	10	34
avg doc length	211.81	211.81
# rel per topic	4398.00	1056.09
% rel per topic	1.56	0.35

Table 3.8: LEGAL dataset statistics

	train	test
# doc per topic	685592	685592
# topic	2	2
avg doc length	470.08	470.08
# rel per topic	505.00	1015.50
% rel per topic	0.10	0.15

Table 3.9: Topic splits of the datasets.

	EMED				TR		LEGAL	
	train	test (EMED 2017)	test (EMED 2018)	test (EMED 2019)	train	test (TR)	train	test (LEGAL)
Topics	CD008686 CD009593 CD011548 CD009372 CD008803 CD009323 CD008691 CD010542 CD009944 CD008760 CD009185 CD009925	CD008081 CD007394 CD007427 CD008054 CD008643 CD008782 CD009020 CD009135 CD009519 CD009551 CD009579 CD009591 CD009647 CD009786 CD010023 CD010173 CD010276 CD010339 CD010386 CD010409 CD010438 CD010632 CD010633 CD010653 CD010705 CD011134 CD011549 CD011975 CD011984 CD012019	CD008122 CD008587 CD008759 CD008892 CD009175 CD009263 CD009694 CD010213 CD010296 CD010502 CD010657 CD010680 CD010864 CD011053 CD011126 CD011420 CD011431 CD011515 CD011602 CD011686 CD011912 CD011926 CD012009 CD012010 CD012083 CD012165 CD012179 CD012216 CD012281 CD012599	CD006468 CD000996 CD001261 CD004414 CD007867 CD009069 CD009642 CD010038 CD010239 CD010558 CD010753 CD011140 CD011571 CD011768 CD011977 CD012069 CD012164 CD012342 CD012455 CD012551 CD012661 CD011558 CD011787 CD008874 CD009044 CD011686 CD012080 CD012233 CD012567 CD012669 CD012768	athome100 athome101 athome102 athome103 athome104 athome105 athome106 athome107 athome108 athome109	401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434	301 302	303 304

3.5.3 Evaluation Metrics

The effectiveness of the proposed framework can be interpreted in three aspects: (1) to maximize the recall, (2) to minimize the assessment cost, and (3) to stop the document selection process on time in the sense that the difference between the achieved recall and the target recall is minimized.

The *recall* metric can be used for the evaluation of (1). It is formally defined as:

$$recall = \frac{r}{R}\,, \tag{3.18}$$

where r is the number of relevant documents identified, R is the total number of relevant documents.

The *cost* metric can be used for the evaluation of (2). It is formally defined as:

$$cost = \frac{n}{N}\,, \tag{3.19}$$

where n is the number of documents shown to the reviewer for assessment, N is the total number of documents for the topic.

The *relative error* between the achieved recall and the target recall can be used for the evaluation of (3). It is formally defined as:

$$RE = \frac{|recall_c - recall_t|}{recall_t}, \tag{3.20}$$

where $recall_t$ denotes the target recall specified by the reviewer.

Following the convention in IR evaluation, it is usually desired to have one metric which considers all the three aspects. However, there does not exist such a metric. As an alternative, we use an existing metric $loss_{er}$ for the combined evaluation of (1) and (2). $loss_{er}$ is introduced for the first time in [37] and latter used as a metric for the total recall task in the CLEF technology assisted reviews in empirical medicine overview tracks [82, 84, 86]. It is defined as:

$$loss_{er} = (100\% - recall_c)^2 + \left(\frac{100}{N}\right)^2 \left(\frac{n}{R + 100}\right)^2, \tag{3.21}$$

where $recall_c$ is the achieved recall when the TAR process is stopped. The intuition behind $loss_{er}$ is straightforward: $100\% - recall_c$ expresses a loss due to the inability to find all relevant documents or to achieve a recall of 100%, and $\frac{n}{R}$ expresses a loss in terms of the effort spent in assessing both relevant and non-relevant documents. Note that the 100% in the formula means that a target recall of 100% is wanted. One may propose to use $recall_t - recall_c$ to replace $100\% - recall_c$ in order to evaluate system performance given different target recall levels, but using $recall_t - recall_c$ penalizes systems that achieve recall higher than the target recall. Also note that the rationale of using $\frac{100}{N}$ and $\frac{n}{R+100}$ can be understood in this way: $aR + b$ (in this case $a = 1, b = 100$) represents a *reasonable* amount of effort to achieve high recall, which has been empirically shown in the TREC total recall tracks [60, 135]; n represents the actual effort spent in reviewing documents; $\frac{n}{R+100}$ actually says how many percentages of the reasonable effort is spent in reviewing documents. $\frac{100}{N}$ is a fixed weight that balances the two types of loss. Readers are also recommended to read the original explanation in [37].

In addition to the aforementioned metrics, it is interesting to see how often the framework achieves a target recall over different topics, because the proposed framework relies on a random sampling of documents. We use *reliability* [37] for the evaluation. *reliability* is defined as the percentage that an estimator achieves the target recall for all topics [37]. It is expressed as:

$$reliability = \frac{|\{q \mid recall_c(q) \leq recall_t(q), q \in \mathcal{Q}|\}}{|\mathcal{Q}|}, \tag{3.22}$$

where $\mathcal{Q}$ is a set of topics, $recall_c(q)$ and $recall_t(q)$ are the achieved recall and the target recall for topic q.

For RQ1, we use all the aforementioned metrics. For RQ2, we compare the difference between R and $\widehat{R}$ through a scatter plot. For RQ3, we interpret the ability

of identifying relevant documents as the gain, and the assessment effort spent in reviewing documents as the loss. We use a curve of *recall* as a function of *cost* for the evaluation. RQ4 is an effectiveness analysis of the submodules of the proposed framework, therefore we use the same metrics as in RQ1.

All the metrics are calculated using open-sourced script `tar_eval`.[12]

3.5.4 Baselines

We compare our approach with several representative baselines, including two greedy methods designed for high recall and early stopping: *Knee* [37] and *Target* [37], one hybrid method of greedy method and sampling method designed for high recall and early stopping: *SCAL* [39], and two variants of *Score Distribution* [71], which are designed for high recall and accurate estimator of R. The *Knee* method is the state-of-the-art method for the total recall task. In what follows, we elaborate on the baseline methods and their precise implementation.

AutoTAR

AutoTAR [35] aims to improve recall by repeatedly selecting documents for users to review. *AutoTAR* is considered the current state-of-the-art method for total recall problems. As it lays the foundation of several methods, we provide the algorithm in Algorithm 3.

We are faced with several design choices in implementing the method. For example, in line 6 of Algorithm 3, when training the ranking model, we are faced with choices such as the corpus and the minimum term frequency used for the TF-IDF vector of documents, and the regularization strength weight used for the Logistic Regression.

We ran our implementation of the *AutoTAR* method against the 30 test topics in the 2017 CLEF technology-assisted reviews in empirical medicine track, as this allows us to compare the results of our implementation with those reported in [40]. We use grid search to sweep over all possible configurations. The corpus varies between the document texts of the whole 30 topics or only of the target topic. The minimum term frequency varies among 2, 3 or 5. The regularization weight varies among 0.001, 1.0, 10000. We find that the best configuration is to use document texts of only the target topic as the corpus, set the minimum term frequency to 2, set the regularization weight to 1.0, which is slightly different from that reported in [40]. We report metrics that are also reported in [40], they are *ap* – average precision, *norm_area* – area under the cumulative recall curve normalized by the optimal area, and *last_rel* – minimum number of documents returned to retrieve all relevant documents. With the best configuration we achieve an *ap* score of 0.191, an *norm_area* score of 0.947, and an *last_rel* score of 493, and the metrics are 0.189, 0.948 and 461 reported in [40]. We use the same configuration for all the latter methods.

Knee

On the basis of the *AutoTAR* method, Cormack et al. [37] propose the *Knee* method. It defines a *knee* of the gain curve through a simple geometric algorithm [140], and

[12] `https://github.com/CLEF-TAR/tar`

Algorithm 3: AutoTAR

Input: Topic q; document collection $\mathcal{C}_q$, target recall $recall_t$.
Output: A ranked list of documents $D_q = [d_1, d_2, \ldots, d_{|\mathcal{C}_q|}]$.
Parameter: None.

t = 0, $\mathcal{L}_0 = \{$pseudo document $d_0\}$, $D_q = []$;
while $\mathcal{C}_q$ is not exhausted **do**
    t += 1;
    Temporarily augment $\mathcal{L}_t$ by uniformly sampling k documents from $\mathcal{U}_t$, labeled non-relevant;
    Train a ranking model on $\mathcal{L}_t$;
    Rank all the documents in $\mathcal{C}_q$ with the ranker trained over $\mathcal{L}_t$;
    Select the top b_t documents from the ranked list and append them in D_q;
    Render relevance assessments for the selected documents;
    Remove the k temporary documents from $\mathcal{L}_t$;
    Place the b_t assessed documents in $\mathcal{L}_t$, and remove them from $\mathcal{U}_t$;
    Update batch size: $b_{t+1} = b_t + [\frac{b_t}{10}]$;
end while

stops the TAR process when the slope after the *knee* diminishes to less than a ratio of the slope before the *knee*. Let $G = \{(x_i, z_i)|i = 1, \ldots, t\}$ denote the data points observed on the grain curve until t-th iteration, x_i denote the cost or percentage of documents reviewed at the i-th iteration, z_i denote the recall at the i-th iteration. Let i^* denote the iteration where the knee is detected. Let ρ denote the slope ratio, defined as $\rho = \frac{|\{d_j|r_{d_j} \leq i^*, y_{d_j}=1\}|}{|\{d_j|t \geq r_{d_j} > i^*\}|} \frac{t-i^*}{i^*}$, where r_{d_j} is the rank of document r_{d_j}, and y_{d_j} is the relevance label of r_{d_j}. If ρ is larger than a bound the TAR process is stopped. We implemented the method described in [37] and summarized it in Algorithm 4.

We run the *Knee* method on the 30 test topics in the 2017 CLEF technology-assisted reviews in empirical medicine track, as it is possible to compare our performance with the performance of *AutoTAR* reported in [40]. As suggested in [37], a dynamic bound, $bound = 156 - \min(R_t, 150)$, has been empirically proven to be effective, where R_t is the number of relevant documents found so far at the t-th iteration. Similar to *AutoTAR*, we also swept over all possible configurations of the ranking model. The best configuration is the same with that of *AutoTAR*. With the best configuration we achieve an *loss_er* score of 0.610 and an *recall* score of 0.999, and the metrics are 0.657, 1.000, reported in [40].

Target

Similar to the *Knee* method, the *Target* method [37] is designed for high recall tasks. It consists of two steps. In the first step it randomly samples documents until a pre-defined number of documents is judged relevant. In the second step, the documents in the collection are ranked and retrieved with the *AutoTAR* method without knowledge of the target set, until each relevant document in the target set has been retrieved. Note that *Target* is designed to achieve high recall, but not 100% recall. It has been strictly

Algorithm 4: Knee

Input: Topic q; document collection $\mathcal{C}_q$, target recall $recall_t$.
Output: A ranked list of documents $D_q = [d_1, d_2, \ldots]$.
Parameter: slope ratio bound: $bound$, number of documents should be reviewed when knee detection takes effect: β.

$D_q = []$, $G = []$;
t = 0, $\mathcal{L}_0 = \{$pseudo document $d_0\}$;
while $\mathcal{C}_q$ is not exhausted & not stop **do**
    t += 1;
    Temporarily augment $\mathcal{L}_t$ by uniformly sampling k documents from $\mathcal{U}_t$, labeled non-relevant;
    Train a ranking model on $\mathcal{L}_t$;
    Rank all the documents in $\mathcal{C}_q$ with the ranker trained over $\mathcal{L}_t$;
    Select the top b_t documents from the ranked list and append them in D_q;
    Render relevance assessments for the selected documents;
    Remove the k temporary documents from $\mathcal{L}_t$;
    Place the b_t assessed documents in $\mathcal{L}_t$, and remove them from $\mathcal{U}_t$;
    Append (x_t, z_t) to G;
    Update batch size $b_{t+1} = b_t + [\frac{b_t}{10}]$;
    if $|\mathcal{L}_t| \geq \beta$ **then**
        if knee is detected & $\rho \geq bound$ **then**
            stop = True;
        end if
    end if
end while

proven that the size of the target set guarantees a recall of 70% with 95% probability under certain assumptions [37].

We implemented the method described in Cormack et al. [37] and summarized it in Algorithm 5.

We ran the *Target* method on the combination of *athome1*, *athome2* and *athome3* datasets, which consists of 30 topics and 290,099 Jeb Bush emails, 465,147 hacker forum documents, and 902,434 local news documents. The datasets are released by the TREC 2015 Total Recall track and results are reported in [37]. There is one hyper-parameter to be tuned: relevant document number in the target set n_{rel}. The recommended value reported in [37] is 10. Based on this we also conducted a hyper-parameter sweeping: we alternate n_{rel} between 5, 10, and 15. The best hyper-parameter configuration ($n_{rel} = 5$) leads to a recall of 0.93 and number of reviewed documents of 49,151, while the reported metrics in [37] are 0.91 and 44,079.

SCAL

SCAL [39] is designed to achieve high recall for large scale or infinite document collection. To this end, it uses a large fixed-sized sample of the collection to generate

Algorithm 5: Target

Input: Target topic q, document collection $\mathcal{C}_q$, target recall $recall_t$.
Output: A ranked list of documents $D = [d_1, d_2, \ldots]$.
Parameter: Relevant document number in the target set: n_{rel}.

1 $D = []$;
2 t = 0, $\mathcal{L}_0 = \{$pseudo document $d_0\}$;
`// Sample set, target set`
3 Sample documents uniformly from $\mathcal{C}_q$ until n_{rel} relevant documents have been found or the collection is exhausted. Let $\mathcal{S}_{sample}$ denote the sampled set, $\mathcal{R}_{sample}$ denote the target set only containing the n_{rel} relevant documents;
4 **if** $\mathcal{C}_q$ is exhausted **then**
5 $\quad D = list(\mathcal{C}_q)$;
6 $\quad$ stop = True;
7 **end if**
`// Re-find relevant documents in the target set`
8 Let $\mathcal{S}_{interact}$ denote documents re-found, $\mathcal{R}_{interact}$ denote the relevant ones. $\mathcal{S}_{interact} = \mathcal{R}_{interact} = \emptyset$;
9 **while** $\mathcal{C}_q$ is not exhausted & not stop **do**
10 $\quad$ t += 1;
11 $\quad$ Temporarily augment $\mathcal{L}_t$ by uniformly sampling k documents from $\mathcal{U}_t$, labeled non-relevant;
12 $\quad$ Train a ranking model on $\mathcal{L}_t$, let f_t denote the corresponding ranking function;
13 $\quad$ Rank all the documents in $\mathcal{C}_q$ with f_t;
14 $\quad$ Select the top b_t documents from the ranked list and add them in $\mathcal{S}_{interact}$;
15 $\quad$ Place the b_t assessed documents in $\mathcal{L}_t$, and remove them from $\mathcal{U}_t$;
16 $\quad$ Remove the k temporary documents from $\mathcal{L}_t$;
17 $\quad b_{t+1} = b_t + [\frac{b_t}{10}]$;
$\quad$ `// Stop the TAR process`
18 $\quad$ **if** $\mathcal{R}_{interact} \subseteq \mathcal{R}_{sample}$ **then**
19 $\quad\quad$ stop = True;
20 $\quad$ **end if**
21 **end while**
`// Final ranked list`
22 Apply f_T on $\mathcal{S}_{sample} \cup \mathcal{S}_{interact}$ to produce a ranked list D, where T denotes the last iteration;

the ranker, estimate the prevalence, and determine the cutoff necessary for a particular target recall.

We implemented the method described in [39] and summarize it in Algorithm 6. *SCAL* solves three major problems. First, in order to deal with the large scale or infinite document collection, a large document set is uniformly sampled (line 3). Second,

Algorithm 6: SCAL

Input: Topic q; document collection $\mathcal{C}_q$, target recall $recall_t$.
Output: A ranked list of documents $D = [d_1, d_2, \ldots]$.
Parameter: Size of the document sample set: N_U, sub-sample size of a batch: b, calibration factor: η.

1 $D = []$;
2 t = 0, $\mathcal{L}_0 = \{$pseudo document $d_0\}, \mathcal{U}_0 = \mathcal{U}, b_0 = 1$;
`// Large sample set`
3 Draw a large uniform random sample set $\mathcal{U}$ of size N_U from $\mathcal{C}_q$;
`// Construct a series of rankers, training sets, and estimators`
4 **while** $\mathcal{U}$ is not exhausted & not stop **do**
5 t += 1;
6 Temporarily augment $\mathcal{L}_t$ by uniformly sampling k documents from $\mathcal{U}_t$, labeled non-relevant;
7 Train a ranking model on $\mathcal{L}_t$, let f_t denote the corresponding ranking function;
8 Rank all the documents in $\mathcal{U}_t$ with f_t;
9 Select the top b_t documents from the ranked list;
10 Calculate the sub-sample size at this iteration: $\dot{b}_t = b$ is $\widehat{R} = 1$ or $b_t < b$, otherwise $\dot{b}_t = b$;
11 Render relevance assessments for the $\dot{b}_t$ documents;
12 Remove the k temporary documents from $\mathcal{L}_t$. Place the $\dot{b}_t$ documents in $\mathcal{L}_t$, and remove the b_t documents from $\mathcal{U}_t$(note $\dot{b}_t$ and b_t are different);
13 $\widehat{R}_{t+1} = \widehat{R}_t + \frac{\dot{r}_t}{\dot{b}_t} b_t$, where $\dot{r}_t$ is the number of relevant documents in the sub-sample;
14 $b_{t+1} = b_t + [\frac{b_t}{10}]$;
15 **end while**
`// Cutoff the ranked list`
16 $\widehat{R} = \eta \cdot \widehat{R}_T$, where T denotes the last iteration;
17 $t^* = \arg\min_t \{\widehat{R}_t \geq recall_t \cdot \widehat{R}\}$;
18 $threshold = \min(\{f_{t^*}(d) \mid d \in \mathcal{U}_0 \backslash \mathcal{U}_{t^*}, d \; is \; relevant\})$;
19 Produce a ranked list $D = \{d \mid f_T(d) \geq threshold, d \in \mathcal{C}_q\}$;

in order to save the effort of reviewing documents, only a finite number instead of the exponentially increasing batch size of documents are sampled and reviewed. The reviewed documents comprise a stratified sample of the entire collection, which is used for training the ranker and calculating $\widehat{R}$ (line 10). Third, it defines a cutoff to stop the TAR process based on the list of rankers and estimators (line 18-20).

Note that line 18 is different from the corresponding line in the original work [39], which is $threshold = \max(\{f_{t^*}(d) \mid d \in \mathcal{U}_0 \backslash \mathcal{U}_{t^*}\})$. During our effort to reproduce

the published results, we found that taking the maximum score over documents in $\mathcal{U}_0 \backslash \mathcal{U}_{t^*}$ will cutoff the final ranked list at a very high rank. When using $\min$ we can achieve similar recall to that reported in [39], but of much larger cost. We further found that using only relevant document scores to determine the threshold can achieve similar recall and cost as in [39].

We ran *SCAL* on the *athome1* dataset, which consists of 10 topics and 290,099 Jeb Bush emails, released with the TREC 2015 Total Recall track, and compare our results to that reported in [39]. There are three hyper-parameters: the size of the large document sample set N_U (%), the sub-sample size of a batch b, the calibration factor η. The recommended configuration reported in [39] is $N_U = 1.0$ (the whole 290,099 emails), $b = 30$, and $\eta = 1.05$. Based on this we conducted a grid search with $N_U \in [0.6, 0.8, 1.0]$, $b \in [30, 50, 70, 90, 110]$, $\eta \in [1.0, 1.05]$. We also compare our line 18 and the responding line in [39]: determining the threshold with either $\max$ or $\min$, filtering either documents in the bucket, or documents in the sampled set, or only relevant documents in the sampled set.

We find that indeed our line 18 leads to similar results as those reported in [39]. The best hyper-parameter configuration ($N_U = 1.0$, $b = 110$, $\eta = 1.05$) leads to recall of 0.91 and number of reviewed documents of 13198, where the metrics are 0.90 and 9504 in [39].

Score Distribution

The score distribution methods proposed by [71] provide ways to estimate the total number of relevant documents at any rank position, making it possible to stop the TAR process at an appropriate cutoff in order to achieve a target recall.

We implement the two methods described in [71] – score distribution using training topics (*SD-training*) and score distribution not using training topics (*SD-sampling*). The main difference is the way of collecting scores of relevant documents. *SD-training* needs extra training topics and the corresponding relevance labels of the documents in the collection, while *SD-sampling* needs to sample documents and render relevance labels from users. The two versions that we implemented are summarized in Algorithm 7 and 8. One notable design choice we are faced with is the ranking model. We use BM25 as the ranking model as it is impossible to use exactly the same ranking model with *AutoTAR*. One reason is that the features used for the logistic regression ranking model of *AutoTAR* are document-wise TF-IDF vectors, and training data is interactively obtained from user relevance feedback; however, the two score distribution methods are not interactive methods, thus extra training topics are needed, and the features must be topic-document-wise if we train a cross-topic ranking model. Therefore, it is not possible to use exactly the same ranking model with *AutoTAR*. Another reason is that what matters for the score distribution methods is the distribution of scores instead of the absolute value of each score [6].

We ran *SD-training* and *SD-sampling* on the 30 test topics in the Technology-Assisted Reviews in Empirical Medicine Task at CLEF 2017, as this makes it possible to compare our implementation with the performance reported in [71]. For *SD-training* we achieved a recall of 0.994 and precision of 0.042, while the metrics reported are 0.989 and 0.057. For *SD-sampling* we fine-tune the hyper-parameter

sample size p_{sample} (%), with $p_{sample} \in [0.1, 0.2, 0.3, 0.4, 0.5]$. The best configuration ($p_{sample} = 0.2$) leads to a recall of 0.978 and precision of 0.062, with Hollmann et al. [71] reporting 0.948 and 0.079.

Algorithm 7: Score distribution using training topics (SD-training)

Input: Target topic q, the corresponding document collection $\mathcal{C}_q$, target recall $recall_t$, training topics, the corresponding document collections and the full relevance judgement sets.
Output: A ranked list of documents $D = [d_1, d_2, \ldots]$.
Parameter: None.

`// Training topics`
1 Produce ranking scores for the relevant documents of all the training topics by applying BM25 ranking function;
2 Fit a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ over the scores of these *relevant* documents;

`// Target topic`
3 Produce ranking scores for all documents of the target topic by applying BM25 ranking function;
4 Find the point where the cumulative density probability of $\mathcal{N}(\mu, \sigma^2)$ equals $1 - recall_t$;
5 Extract from $\mathcal{C}_q$ all the documents of which the ranking scores are larger than the point;

`// Final ranked list`
6 Sort the extracted documents by the descending order of ranking scores, denoted as D;

3.5.5 Implementation

Similar to *AutoTAR* and related methods, our proposed framework is topic-wise independent, which means a brand new ranking model is trained at the beginning of the TAR process for each topic.

For fair comparison in terms of precisely stopping the TAR process at a target recall, we use the same document representation and ranking model as *AutoTAR*. Specifically, we use TF-IDF document representation and logistic regression for ranking: we consider all documents associated to a given topic as the corpus, and construct a TF-IDF vector for each document as its representation. We use *scikit-learn* (a Python Machine Learning toolkit)[13] for the implementation. In particular, we use `TfidfVectorizer` with its default configuration to preprocess the input text and generate the TF-IDF vector, while we use `LogisticRegression` with the default configuration (which is also proven to be the best configuration when reproducing the *AutoTAR* baseline) for the logistic regression models.

We also implemented and reproduced all the baselines: *AutoTAR*, *Knee*, *Target*,

[13] `https://scikit-learn.org/`

Algorithm 8: Score distribution not using training topics (SD-sampling)

Input: Target topic q, the corresponding document collection $\mathcal{C}_q$, target recall $recall_t$.
Output: A ranked list of documents $D = [d_1, d_2, \ldots]$.
Parameter: Sample size p_{sample} (%).

```
// Sample documents
```
1 Produce a ranked list of all the documents and a ranked list of normalized score by applying BM25 ranking function on $\mathcal{C}_q$;
2 Sample uniformly $|\mathcal{C}_q| \cdot p_{sample}$ documents, denoted by $\mathcal{D}_1$;

```
// Fit Gaussian distribution
```
3 Fit a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ over the scores of the sampled *relevant* documents;
4 Find the point where the cumulative density probability of $\mathcal{N}(\mu, \sigma^2)$ equals $1 - recall_t$;
5 Extract from $\mathcal{C}_q$ the documents of which the ranking scores are larger than the point, denoted by $\mathcal{D}_2$;

```
// Final ranked list
```
6 Output the final ranked list D by merging $\mathcal{D}_1$ and $\mathcal{D}_2$ by the descending order of ranking scores;

SCAL, *SD-training* and *SD-sampling* given that their source code is not available online. The code used in this chapter will be released as open source code online.[14]

3.6 Results and Analysis

In this section, we conduct experiments on all the datasets to answer the aforementioned research questions.

3.6.1 Stopping Effectiveness

This experiment is designed to answer RQ1. In this experiment, methods to be compared are provided with a target recall. A method is considered better than another whether (1) it achieves higher recall, (2) it spends less assessment cost, and (3) the achieved recall when it decides to stop the TAR process is as close as possible to the target recall. As explained in Section 3.5.3, we use a portfolio of metrics including $recall$, $cost$, RE, $loss_{er}$, and $reliability$ for evaluation.

Experimental Setting

As our framework provides the flexibility to target the TAR process for *recall*, we vary the *target recall* value between 0.8, 0.9, and 1.0 to see whether the proposed framework is capable of stopping the TAR process on time. We focus on $\{0.8, 0.9, 1.0\}$ because high recall is usually desired in the TAR process.

[14] https://github.com/dli1/auto-stop-tar

The proposed framework provides multiple choices for its sampling module, estimation module and stopping module. The estimator varies between the Horvitz-Thompson estimator and the Hansen-Hurwitz estimator, and the stopping strategy varies between *optimistic* and *conservative*. In total there are five variations of the proposed framework: HT-opt, HT-con1 (variance is calculated based on Equation 3.6), HT-con2 (variance is calculated based on Equation 3.7), HH-opt, HH-con (variance is calculated based on Equation 3.9).

By running the five variations on the corresponding training topics of the EMED, TR and LEGAL datasets, we find the combination of APPrior and HT-con1 performs the best. In the rest of the chapter, we use this configuration as our method. The impact on the stopping effectiveness with regards to the sampling methods (APPrior, PowerLaw, MixtureUniform), estimators (HT and HH) and stopping strategies (optimistic and conservative) can be found in Section 3.6.4.

This experiment is conducted on all five test datasets including EMED 2017, EMED 2018, EMED 2019, TR and LEGAL. Note that for a topic on TR and LEGAL, there is no associated documents filtered with some initial ranking techniques like on the EMED datasets, therefore we use the whole document collection as the associated documents for each topic. On the other hand, our method typically needs a large amount of memory (or a long time depending on the implementation) when calculating the first order and the second order inclusion probabilities. For example, it needs about 20GB memory to deal with topics associated with about 15,000 documents. As a consequence it is impossible to run our method directly on our computational resources. As an alternative, for the TR and LEGAL datasets, we split the whole collection into buckets of 1000 documents, run our method over these buckets, and concatenate all the sampled documents for the final reviewing. This adaption is only for our method. All the baselines are run on the TR and LEGAL dataset as it is.

We compare our method with *Knee*, *Target*, *SCAL*, *SD-training* and *SD-sampling*. The *Knee* method and the *Target* method are designed to achieve 100% recall, and the rest are designed to stop the TAR process at a specified target recall. We present the results of the *Knee* method and the *Target* method only when the target recall is 1.0. We fine-tune all the methods (if needed) on the training topics when the target recall is 0.8, 0.9 and 1.0, respectively. The *Knee* method has two hyper-parameters: slope ratio bound: $bound$, number of reviewed documents when knee detection takes effect: β. The recommended value reported in [37] is $bound = 156 - min(R_t, 150)$ and $\beta = 1000$. Based on this we conducted a grid search for $bound \in [3, 6, 10, 156 - min(R_t, 150)]$, and $\beta \in [100, 1000]$. The *Target* method has one hyper-parameter: relevant document number in the target set n_{rel}. The recommended value reported in [37] is $n_{rel} = 10$. Based on this, we conducted a grid search by alternating $n_{rel} = [5, 10, 15]$. *SCAL* has three hyper-parameters: the size of the large document sample set N_U (%), the sub-sample size of a batch b, the calibration factor η. The recommended configuration reported in [39] is $N_U = 1.0$, $b = 30$, and $\eta = 1.05$. Based on this we conducted a grid search for $N_U \in [0.6, 0.8, 1.0]$, $b \in [30, 50, 70, 90, 110]$, $\eta \in [1.0, 1.05]$. *SD-training* has no hyper-parameter. *SD-sampling* has one hyper-parameter: sample size $p_{sample} \in [0.1, 0.2, 0.3, 0.4, 0.5]$. The best configuration of these baselines for different target recall is shown in Table 3.10.

Table 3.10: Fine-tuned parameters.

target recall	Knee	β	Target	SCAL			SD-training	SD-sampling
	$bound$		n_{rel}	N_U	b	η	-	p_{sample}
EMED 2017/2018/2019								
0.8	$156 - \min(R_t, 150)$	1000	15	0.8	50	1.05	-	0.4
0.9	$156 - \min(R_t, 150)$	1000	15	0.8	70	1.05	-	0.3
1.0	$156 - \min(R_t, 150)$	1000	15	1.0	110	1.05	-	0.2
TR								
0.8	10	1000	15	0.8	70	1.05	-	0.1
0.9	10	1000	15	1.0	110	1.05	-	0.1
1.0	10	1000	15	1.0	110	1.05	-	0.1
LEGAL								
0.8	$156 - \min(R_t, 150)$	1000	10	1.0	30	1.05	-	0.1
0.9	$156 - \min(R_t, 150)$	1000	10	1.0	30	1.05	-	0.1
1.0	$156 - \min(R_t, 150)$	1000	10	1.0	30	1.05	-	0.1

Table 3.11: Stopping effectiveness ($recall_t = 1.0$).

Method	$recall_c \uparrow$	$cost \downarrow$	$reliability \uparrow$	$loss_{er} \downarrow$	$RE \downarrow^*$
			EMED 2017		
Knee	0.998 ± 0.006	**0.291 ± 0.194**	0.833 ± 0.379	**0.041 ± 0.081**	0.002 ± 0.006
Target	0.978 ± 0.036	0.614 ± 0.258	0.567 ± 0.504	0.292 ± 0.349	0.022 ± 0.036
SCAL	0.984 ± 0.038	0.659 ± 0.281	0.700 ± 0.466	0.253 ± 0.265	0.016 ± 0.038
SD-training	**0.999 ± 0.003**	0.997 ± 0.007	**0.967 ± 0.183**	0.466 ± 0.292	0.010 ± 0.001
SD-sampling	0.973 ± 0.042	0.762 ±0.283	0.533 ± 0.507	0.311 ± 0.320	0.027 ± 0.036
Ours	**0.999 ± 0.008**	0.625 ± 0.190	**0.967 ± 0.183**	0.161 ± 0.115	**0.001 ± 0.008**
			EMED 2018		
Knee	0.994 ± 0.013	**0.328 ± 0.258**	0.700 ±0.466	**0.047 ± 0.080**	**0.006 ± 0.013**
Target	0.983 ± 0.022	0.594 ± 0.289	0.500 ± 0.509	0.251 ± 0.310	0.017 ± 0.022
SCAL	0.982 ± 0.039	0.657 ± 0.284	0.600 ± 0.498	0.233 ± 0.254	0.018 ± 0.039
SD-training	**1.000 ± 0.000**	0.996 ± 0.008	**1.000 ± 0.000**	0.394 ± 0.277	0.010 ± 0.000
SD-sampling	0.964 ± 0.093	0.688 ± 0.270	0.533 ± 0.507	0.180 ± 0.170	0.036 ± 0.090
Ours	0.992 ± 0.044	0.662 ± 0.182	0.967 ± 0.183	0.163 ± 0.136	0.008 ± 0.044
			EMED 2019		
Knee	0.998 ± 0.009	**0.420 ± 0.311**	0.968 ± 0.180	**0.122 ± 0.160**	0.002 ± 0.009
Target	0.985 ± 0.025	0.753 ± 0.275	0.645 ± 0.486	0.428 ± 0.350	0.015 ± 0.025
SCAL	0.993 ± 0.022	0.808 ±0.219	0.839 ± 0.374	0.410 ± 0.288	0.007 ± 0.022
SD-training	0.999 ± 0.004	0.992 ± 0.016	0.968 ± 0.180	0.545 ± 0.262	0.010 ± 0.001
SD-sampling	0.974 ± 0.070	0.767 ± 0.249	0.623 ± 0.489	0.306 ± 0.273	0.028 ± 0.066
Ours	**1.000 ± 0.000**	0.651 ± 0.198	**1.000 ± 0.000**	0.223 ± 0.148	**0.000 ± 0.000**
			TR		
Knee	0.960 ± 0.056	**0.016 ± 0.041**	0.088 ± 0.288	**0.005 ± 0.016**	0.040 ± 0.056
Target	0.944 ± 0.063	0.120 ± 0.150	0.147 ± 0.359	0.024 ± 0.068	0.056 ± 0.063
SCAL	0.919 ± 0.168	0.146 ± 0.317	0.147 ± 0.359	0.079 ± 0.211	0.081 ± 0.168
SD-training	1.000 ± 0.000	1.000 ± 0.000	**1.000 ± 0.000**	0.122 ± 0.173	0.010 ± 0.000
SD-sampling	0.988 ± 0.066	0.944 ± 0.225	0.941 ± 0.239	0.106 ± 0.142	0.022 ± 0.063
Ours	**1.000 ± 0.000**	0.779 ± 0.148	0.941 ± 0.239	0.100 ± 0.160	**0.000 ± 0.000**
			LEGAL		
Knee	0.966 ± 0.048	**0.287 ± 0.272**	0.500 ± 0.707	0.004 ±0.002	0.034 ± 0.048
Target	0.953 ± 0.028	0.147 ± 0.008	0.000 ± 0.000	**0.003 ± 0.003**	0.047 ± 0.028
SCAL	0.039 ± 0.031	0.005 ± 0.001	0.0 ± 0.0	0.924 ± 0.060	0.961 ± 0.031
SD-training	**1.000 ± 0.000**	1.000 ± 0.000	**1.000 ± 0.000**	0.007 ± 0.002	0.010 ± 0.000
SD-sampling	**1.000 ± 0.000**	1.000 ± 0.000	**1.000 ± 0.000**	0.007 ± 0.002	0.010 ± 0.000
Ours	0.996 ± 0.001	0.833 ± 0.025	0.000 ± 0.000	0.005 ± 0.001	**0.004 ± 0.001**

*: ↑ means that higher values are better, and ↓ means that lower values are better.

Table 3.12: Stopping effectiveness ($recall_t = 0.9$).

Method	$recall_c (\approx recall_t)^*$	$cost \downarrow$	$reliability \uparrow$	$loss_{er} \downarrow$	$RE \downarrow$
			EMED 2017		
SCAL	**0.914 ± 0.075**	0.496 ± 0.244	0.667 ± 0.479	0.168 ± 0.209	0.072 ± 0.042
SD-training	0.955 ± 0.057	0.691 ± 0.054	**0.833 ± 0.379**	0.233 ± 0.148	0.080 ± 0.034
SD-sampling	**0.902 ± 0.083**	0.506 ± 0.277	0.567 ± 0.504	0.192 ± 0.278	0.071 ± 0.057
Ours	**0.884 ± 0.088**	**0.421 ± 0.097**	0.500 ± 0.509	**0.097 ± 0.065**	**0.069 ± 0.070**
			EMED 2018		
SCAL	**0.902 ± 0.087**	0.493 ± 0.241	0.667 ± 0.479	0.154 ± 0.168	0.074 ± 0.060
SD-training	0.972 ± 0.033	0.701 ± 0.038	**0.967 ± 0.183**	0.196 ± 0.138	0.082 ± 0.030
SD-sampling	0.855 ± 0.108	**0.379 ± 0.217**	0.367 ± 0.490	**0.077 ± 0.074**	0.080 ± 0.102
Ours	**0.892 ± 0.075**	0.441 ± 0.103	0.600 ± 0.498	0.098 ± 0.078	**0.046 ± 0.071**
			EMED 2019		
SCAL	**0.893 ± 0.104**	0.621 ± 0.206	0.516 ± 0.508	0.271 ± 0.198	0.082 ± 0.082
SD-training	0.940 ± 0.100	0.713 ± 0.043	**0.774 ± 0.425**	0.295 ± 0.156	0.092 ± 0.075
SD-sampling	**0.893 ± 0.095**	0.517 ± 0.270	0.508 ± 0.504	0.198 ± 0.260	0.072 ± 0.077
Ours	**0.878 ± 0.096**	**0.479 ± 0.129**	0.387 ± 0.495	**0.159 ± 0.154**	**0.072 ± 0.080**
			TR		
SCAL	**0.903 ± 0.171**	**0.144 ± 0.318**	0.647 ± 0.485	**0.083 ± 0.210**	0.094 ± 0.163
SD-training	1.000 ± 0.000	1.000 ± 0.000	**1.000 ± 0.000**	0.122 ± 0.173	0.111 ± 0.000
SD-sampling	0.936 ± 0.129	0.779 ± 0.407	0.794 ± 0.410	0.102 ± 0.136	0.133 ± 0.063
Ours	0.953 ± 0.030	0.766 ± 0.163	0.941 ± 0.239	0.103 ± 0.158	**0.062 ± 0.027**
			LEGAL		
SCAL	0.039 ± 0.031	0.005 ± 0.001	0.000 ± 0.000	0.924 ± 0.060	0.957 ± 0.035
SD-training	1.000 ± 0.000	1.000 ± 0.000	**1.000 ± 0.000**	**0.007 ± 0.002**	0.111 ± 0.000
SD-sampling	1.000 ± 0.000	1.000 ± 0.000	**1.000 ± 0.000**	**0.007 ± 0.002**	0.111 ± 0.000
Ours	**0.803 ± 0.006**	**0.811 ± 0.029**	0.000 ± 0.000	0.043 ± 0.004	**0.108 ± 0.007**

*: $recall_c$ should be as close as $recall_t$.

Table 3.13: Stopping effectiveness ($recall_t = 0.8$).

Method	$recall_c$ ($\approx recall_t$)*	$cost \downarrow$	$reliability \uparrow$	$loss_{er} \downarrow$	$RE \downarrow$
			EMED 2017		
SCAL	0.888 ± 0.089	0.451 ± 0.255	**0.800 ± 0.407**	0.177 ± 0.246	0.138 ± 0.072
SD-training	0.881 ± 0.113	0.417 ± 0.068	0.767 ± 0.430	0.109 ± 0.062	0.148 ± 0.088
SD-sampling	**0.798 ± 0.087**	**0.350 ± 0.270**	0.433 ± 0.504	0.170 ± 0.269	**0.077 ± 0.076**
Ours	**0.787 ± 0.090**	**0.335 ± 0.077**	0.367 ± 0.490	**0.105 ± 0.056**	0.088 ± 0.080
			EMED 2018		
SCAL	0.862 ± 0.093	0.428 ± 0.245	0.767 ± 0.430	0.144 ± 0.162	0.119 ± 0.070
SD-training	0.886 ± 0.094	0.414 ± 0.059	**0.833 ± 0.379**	**0.086 ± 0.055**	0.141 ± 0.074
SD-sampling	0.753 ± 0.137	**0.258 ± 0.168**	0.367 ± 0.490	0.099 ± 0.100	0.121 ± 0.134
Ours	**0.781 ± 0.073**	0.347 ± 0.089	0.467 ± 0.507	0.104 ± 0.061	**0.064 ± 0.069**
			EMED 2019		
SCAL	0.887 ± 0.086	0.577 ± 0.245	**0.903 ± 0.301**	0.261 ± 0.228	0.134 ± 0.071
SD-training	0.826 ± 0.153	0.421 ± 0.066	0.613 ± 0.495	**0.146 ± 0.085**	0.155 ± 0.114
SD-sampling	**0.787 ± 0.125**	**0.366 ± 0.249**	0.475 ± 0.504	0.166 ± 0.217	0.111 ± 0.110
Ours	**0.791 ± 0.121**	**0.397 ± 0.119**	0.452 ± 0.506	0.158 ± 0.143	**0.111 ± 0.100**
			TR		
SCAL	0.761 ± 0.288	**0.107 ± 0.282**	0.676 ± 0.475	0.167 ± 0.285	0.247 ± 0.263
SD-training	1.000 ± 0.000	1.000 ± 0.000	**1.000 ± 0.000**	0.122 ± 0.173	0.250 ± 0.000
SD-sampling	0.896 ± 0.168	0.690 ± 0.456	0.735 ± 0.448	**0.100 ± 0.117**	0.231 ± 0.063
Ours	**0.885 ± 0.053**	0.754 ± 0.174	0.912 ± 0.288	0.115 ± 0.153	**0.111 ± 0.056**
			LEGAL		
SCAL	0.039 ± 0.031	**0.005 ± 0.001**	0.000 ± 0.000	0.924 ± 0.060	0.952 ± 0.039
SD-training	**1.000 ± 0.000**	1.000 ± 0.000	**1.000 ± 0.000**	**0.007 ± 0.002**	0.250 ± 0.000
SD-sampling	**1.000 ± 0.000**	1.000 ± 0.000	**1.000 ± 0.000**	**0.007 ± 0.002**	0.250 ± 0.000
Ours	0.684 ± 0.022	0.794 ± 0.032	0.000 ± 0.000	0.105 ± 0.015	**0.145 ± 0.027**

*: $recall_c$ should be as close as $recall_t$.

Results

Effectiveness in terms of high recall and low cost. Achieving 100% recall is desired in electronic discovery, systematic review, investigation, research, and the construction of datasets for information retrieval evaluation. It is interesting to see whether the stopping methods can achieve 100% recall with low cost.

In Table 3.11 the target recall is set to 100%, and we focus on the $recall$, $cost$ and $loss_{er}$ values. On the EMED 2017 dataset, all the methods can achieve very high recall (97.8% to 99.9%); among them *Knee* only needs around 30% cost, while *Target*, *SCAL*, *SD-sampling* and our method needs double cost (61.4% to 76.2%), and *SD-training* needs to review almost the full document collection. When comparing the $loss_{er}$ value which considers both $recall$ and $cost$, *Knee* achieves the best performance, and our method achieves the second best performance. Similar results can be observed on the EMEM 2018, EMED 2019 and TR datasets. Note that the performance of all methods on LEGAL is very different from those on the remaining four datasets. The cost on TR an LEGAL are always larger than that on EMED. This is because our method applied on TR and LEGAL is a slightly different from the original algorithm, as mentioned at the beginning of this section. Specific reason is that we train a new ranking model from scratch for each split of the document collection, which leads to reviewing more non-relevant documents. This issue can be addressed by training a global ranking model for all the splits, we leave this for the future work.

In order to better understand the relation of $cost$ and $recall$, we further visualize the $cost$ and $recall_c$ columns of Table 3.11. Figure 3.3 shows the scatter plots of different methods, and the $cost$ and $recall_c$ values are averaged over all topics. When the target recall is set to 100% (subplots (a) to (e) in Figure 3.3), the *Knee* method, the *SD-training* method and our method are on the Pareto frontier, indicating they achieve

a good balance between cost and recall.

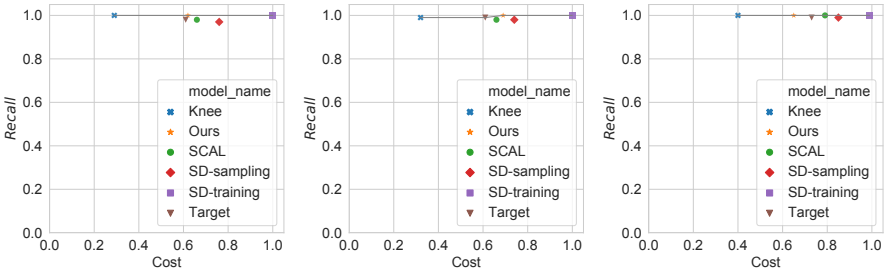
Overall, when the goal is achieving high recall and low cost, the *Knee* method performs the best. It is a greedy method designed for total recall tasks; there is no sampling mechanism and the goal is to collect many relevant documents as fast as possible. Our method achieves the second best performance. It is a sampling method that trades off the assessment cost for the estimation of R.

Effectiveness in terms of on-time stopping. Another goal of the proposed framework is to stop the TAR process on time. It is also interesting to see whether the recall achieved when the TAR process is stopped is close to the target recall and how often the achieved recall is equal to or higher than the target recall. We vary target recall value between 0.8, 0.9 and 1.0 in Table 3.11, 3.12 and 3.13, and focus on the RE and $reliability$ values.

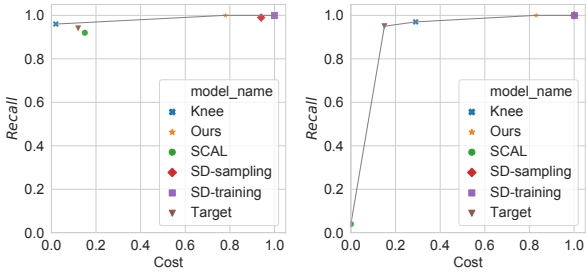
First, let us examine each table and compare the performance of different methods. Our method achieves either the best or comparable RE and $reliability$ values among all baselines, indicating that it can stop the TAR process on time. The *Knee* method performs slightly worse than our method, because it stops the TAR process early and thus it is more likely to achieve a recall less than 100%. The *Target* method performs worse than the *Knee* method because it needs much more assessment cost while still achieving slightly lower recall than the *Knee* method; the observation is similar with the conclusion in [37]. Note that given a target recall, although it is possible to fine-tune the hyper-parameters for the *Knee* method, or adapt the method by collecting a subset of the target set for the *Target* method, these tricks do not represent their original intentions. Therefore we only report the results when $recall_t = 1.0$ for *Knee* and *Target*. The *SCAL* method, which also stops the TAR process based on an estimator of R, can achieve the target recall accurately but the cost is higher than *Knee* as expected. The *SD-training* method estimates the cutoff of stopping reviewing documents by fitting a Gaussian distribution of the ranking scores of relevant documents of training data. It is not an interactive method. Instead, the documents are only ranked one time. *SD-training* does not stop the TAR process on time and it also needs more cost. *SD-sampling* is similar to *SD-training*, the difference is that the documents which provide ranking scores are sampled from the documents of the current topic, instead of training topics. *SD-sampling* stops the TAR process better than *SD-training*.

Now, let us examine how the RE and $reliability$ values change across the three tables. It can be found that the RE and $reliability$ values of all the methods increase when the target recall decreases. *SCAL* stops the TAR process on time when target recall is 1.0 and 0.9, but not on time when target recall is 0.8. *SD-sampling* stops the TAR process more accurately than *SD-training*, indicating sampling relevant documents within one topic is effective for on-time stopping. The aforementioned observations are also supported in the scatter plots in Figure 3.3 and 3.4 (values averaged over topics), Figure 3.4 and 3.4 (topic-wise values). The desirable situation is to stop the TAR process exactly when $recall_c = recall_t$.

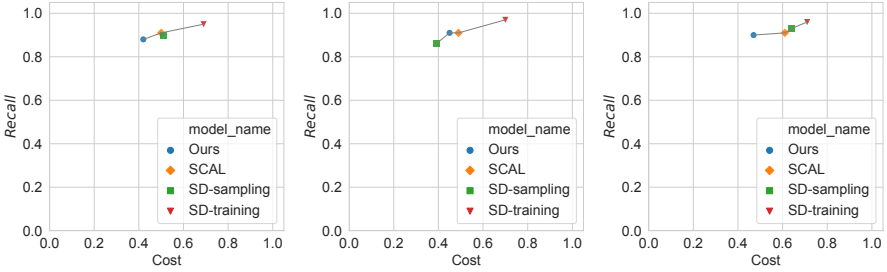
Overall, when the goal is to stop the TAR process on time, our method performs better than *Knee*. *SCAL*, *SD-sampling* also performs comparably well, but *SD-training* does not perform as well as them.



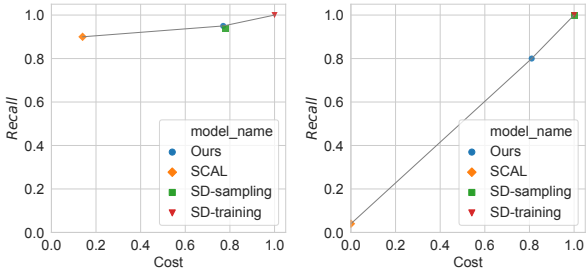
(a) EMED 2017, $recall_t = 1.0$ (b) EMED 2018, $recall_t = 1.0$ (c) EMED 2019, $recall_t = 1.0$



(d) TR, $recall_t = 1.0$ (e) LEGAL, $recall_t = 1.0$

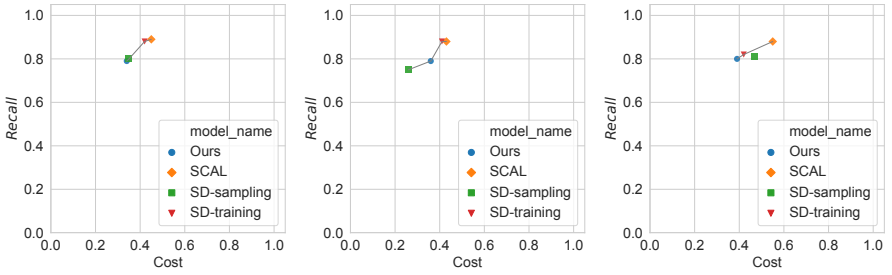


(f) EMED 2017, $recall_t = 0.9$ (g) EMED 2018, $recall_t = 0.9$ (h) EMED 2019, $recall_t = 0.9$

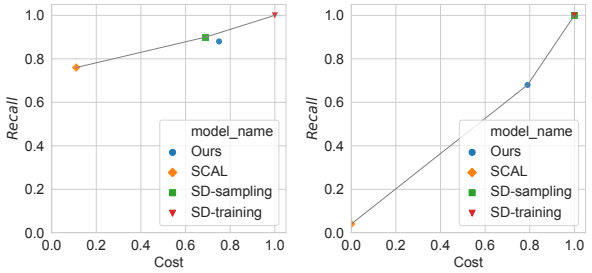


(i) TR, $recall_t = 0.9$ (j) LEGAL, $recall_t = 0.9$

Figure 3.3: Visualization of stopping effectiveness on five datasets. x-axis is *cost*, y-axis is $recall_c$. Each point in the plots represents one method. The desirable situation is to stop the TAR process exactly when $recall_c = recall_t$.



(a) EMED 2017, $recall_t = 0.8$ (b) EMED 2018, $recall_t = 0.8$ (c) EMED 2019, $recall_t = 0.8$



(d) TR, $recall_t = 0.8$ (e) LEGAL, $recall_t = 0.8$

Figure 3.3: (Con't) Visualization of stopping effectiveness on five datasets. x-axis is $cost$, y-axis is $recall_c$. Each point represents one method. The desirable situation is to stop the TAR process exactly when $recall_c = recall_t$.

3.6.2 Estimating R

This experiment is designed to answer RQ2. We examine whether the estimator $\widehat{R}$ is unbiased with low variance. Further, as the estimator $\widehat{R}$ is expected to be more accurate with more sampled documents, we also examine the estimator $\widehat{R}$ changes as the target recall changes.

Experimental setting

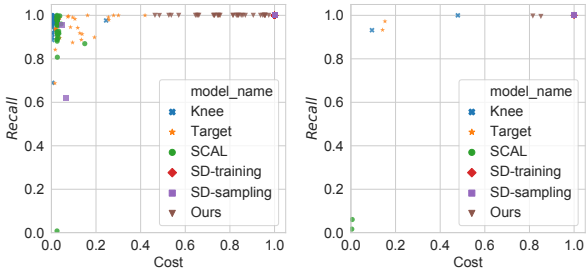
The experimental setting is the same as in the previous experiment. We only report results for the EMED 2017 dataset since the results for the other datasets are similar. Because *Knee*, *Target*, *SD-training* and *SD-sampling* do not provide an estimation of R, we only compare our method with *SCAL*.

Results

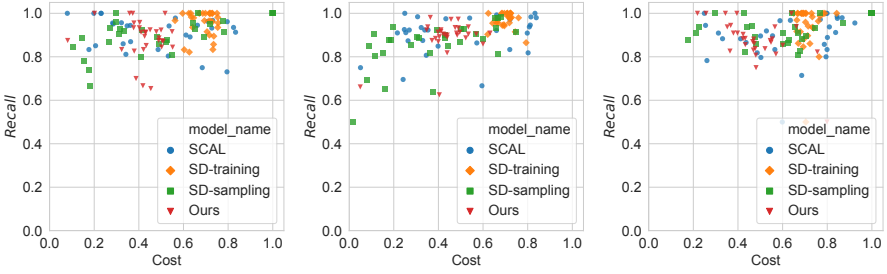
For each topic, we record R and $\widehat{R}$ when the TAR process stops and present them in scatter plots shown in Figure 3.5. We also report the MSE metric to see how far the estimated value is from the true value.

We can see that most points of our method lie on the diagonal line when $recall_t = 1.0$, indicating that our method accurately estimate R. When $recall_t = 0.9$ and $recall_t = 0.8$ R is under-estimated. But if we also take $\widehat{var}(\widehat{R})$ into consideration,

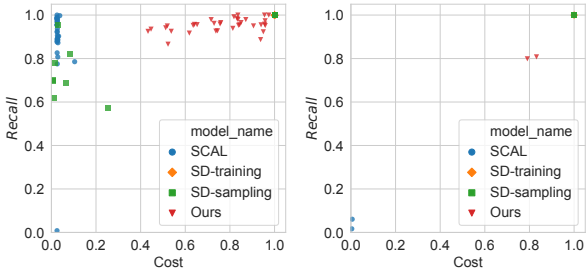
(a) EMED 2017, $recall_t = 1.0$ (b) EMED 2018, $recall_t = 1.0$ (c) EMED 2019, $recall_t = 1.0$



(d) TR, $recall_t = 1.0$ (e) LEGAL, $recall_t = 1.0$

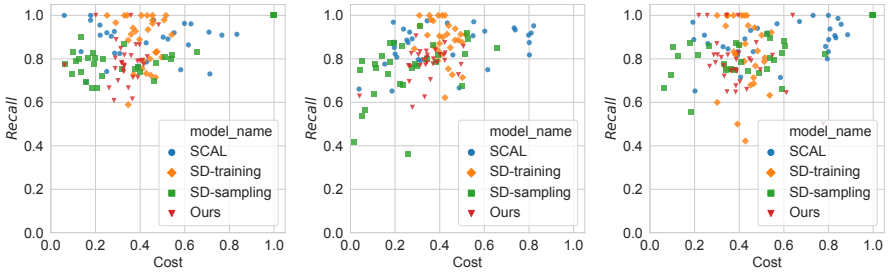


(f) EMED 2017, $recall_t = 0.9$ (g) EMED 2018, $recall_t = 0.9$ (h) EMED 2019, $recall_t = 0.9$

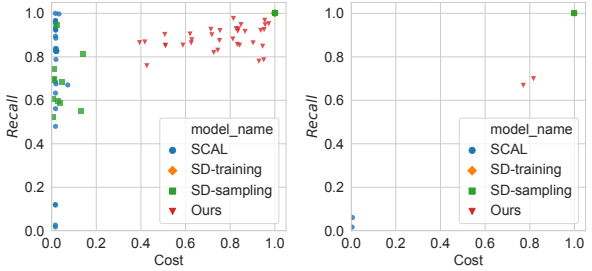


(i) TR, $recall_t = 0.9$ (j) LEGAL, $recall_t = 0.9$

Figure 3.4: Topic-wise visualization of stopping effectiveness on five datasets. Color and marker together distinguish different methods. Each point represents one topic.



(a) EMED 2017, $recall_t = 0.8$ (b) EMED 2018, $recall_t = 0.8$ (c) EMED 2019, $recall_t = 0.8$



(d) TR, $recall_t = 0.8$ (e) LEGAL, $recall_t = 0.8$

Figure 3.4: (Con't) Topic-wise visualization of stopping effectiveness on five datasets. Color and marker together distinguish different methods. Each point represents one topic.

i.e. the bar of one standard deviation $\widehat{R} \pm \widehat{var}(\widehat{R})$, it is more likely that the bar covers R when $recall_t = 0.9$ and $recall_t = 0.8$. The major reason of under-estimation for low target recall are due to the low prevalence of the topics in the collection. At the early iterations the estimator fluctuates drastically. With more documents, especially more non-relevant document for low-prevalence topic being sampled, the inclusion probabilities are generally approaching to 1.0. Hence, finding a new relevant document will only slightly increase $\widehat{R}$ and this makes the estimator slowly approaching the true value. It is interesting to study how the accuracy of the estimator changes with different prevalence values. We leave this in the future work.

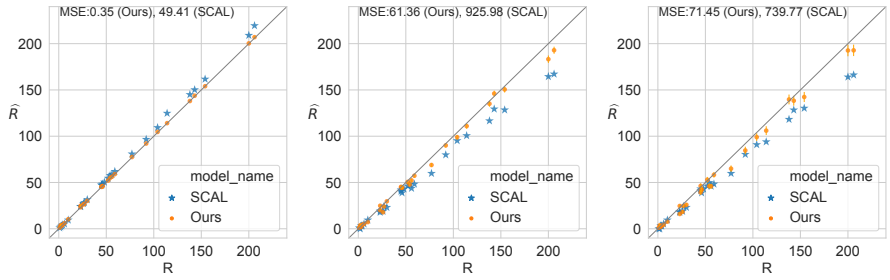
The *SCAL* baseline method also under-estimates R for $recall_t = 0.9$ and $recall_t = 0.8$ but over-estimate R for $recall_t = 1.0$.

Overall, the estimator $\widehat{R}$ of our method is accurate especially for task of high target recall and summing up $\widehat{var}(\widehat{R})$ and $\widehat{R}$ partially solves the under-estimation problem for task of low target recall.

3.6.3 Trade off Recall against Estimating R

Experimental setting

This experiment is designed to answer RQ3. The motivation for this research question is that we introduce extra assessment costs when we use random sampling instead of a greedy method. Hence, it is beneficial to know whether we sacrifice too much the effectiveness of the ranker to collect relevant documents for the ability to accurately



(a) $recall_t = 1.0$ (b) $recall_t = 0.9$ (c) $recall_t = 0.8$

Figure 3.5: R v.s. $\widehat{R}$ on EMED 2017 (topic-wise). In each subplot, marker and color distinguish estimators; each point corresponds to a topic. The bar on each point indicates a one standard deviation ($\sqrt{\widehat{var}(\widehat{R})}$) away from mean ($\widehat{R}$). An unbiased estimator with zero variance should lead to points that lie on the $x = y$ line.

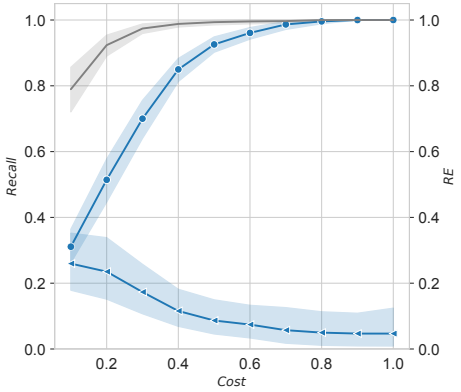


Figure 3.6: Trade off recall against estimating R. The dataset is EMED 2017. The grey line (*AutoTAR*) serves as an upper bound of the recall. The blue dotted line and the blue triangle line indicates how much $recall$ our method trades off against accurately estimating $\widehat{R}$.

estimate the number of missing relevant documents through sampling. In the previous experiment, we study the quality of the estimator $\widehat{R}$ when the TAR process stops. This is a "static" experiment in the sense that it only considers $\widehat{R}$ at the final iteration. In this experiment, we will show how the actual $recall$ and the *relative error* between R and $\widehat{R}$ changes along the iteration. For each topic, we let the TAR process run until it almost exhausts all the documents, reaching at a $cost$ of 95%.

Results

Figure 3.6 shows the tradeoff between high recall and accurate estimation of R. The *AutoTAR* method (the grey line) serves as an upper bound of recall. An interesting point is when *AutoTAR* achieves a recall of 100% with a cost of 40%, our method

achieves a recall of 85% and also provides an estimation of R with a relative error of 10%.

The mechanism of *AutoTAR* is to repeatedly select documents from the top of the ranking towards the bottom until all the documents are selected; is does not address the stopping problem. Built on top of the *AutoTAR*, the *Knee* method allows for automatic stopping by using a geometric algorithm to detect the knee of the recall curve. It does not need extra assessment cost to determine the stopping point, however, it does not provide any insight on how many relevant documents one would miss if one stops reviewing. It turns out that knowing this information does need to trade off a certain amount of recall. The overall finding is that we need to trade off around 15% relevant documents against estimating R with around a relative error of 10%.

Knowing an estimation of R is necessary in many cases, for example, estimating the number of missing documents to study the reliability of the results of systematic reviews [68], estimating the volume of social media posts to track brands popularity and product sales [181]. The framework can be applied in these application fields.

3.6.4 Model Component Analysis

Experimental setting

This experiment is designed to answer RQ4. We examine how the Horvitz-Thompson (HT) and Hansen-Hurwitz (HH) estimators, and the *optimistic* (opt) and *conservative* (con) stopping strategies impact the performance of the proposed framework. The combination of the estimators and the stopping strategies give rise to the following options: HT-opt, HT-con1 (where the variance is calculated based on Equation (3.6)), HT-con2 (where the variance is calculated based on Equation (3.7)), HH-opt, and HH-con1 (where the variance is calculated based on Equation (3.9)). We conduct this experiment on the training topics of EMED 2017. Similar with the previous setting, we alternate the target recall level between 0.8, 0.9, and 1.0.

Results

We show the performance of each configuration with respect to $loss_{er}$ and RE in Table 3.14.

Estimator. The Horvitz-Thompson estimator performs slightly better than the Hansen-Hurwitz estimator in terms of both $loss_{er}$ and RE. In practice, if the goal is to achieve high effectiveness, the Horvitz-Thompson estimator is recommended. Note that the calculation of the Horvitz-Thompson estimator is of relative low-efficiency, and the way of calculating inclusion probability is not trivial if the sampling is not random sampling with replacement. In other words, if efficiency is a concern or the sampling is without replacement, the Hansen-Hurwitz estimator is recommended.

Stopping strategy. The *conservative* strategy performs slightly better than the *optimistic* strategy. In practice, if not missing relevant documents is very important and the cost is not the main concern, the *conservative* strategy is recommended. We find that in our experimental results, the estimator $\widehat{R}$ tends to be lower than the true value R (as shown in Figure 3.5), therefore adding one standard deviation (remember that both the mean and variance are provided in the estimation module) to the estimated

mean value does help to stop the TAR process more effectively. In all the experiments in this chapter, we have only tried to add one standard deviation; adding two or three standard deviations can be tried in order to mitigate the underestimation of R in future work.

Table 3.14: Impact of the Hansen-Hurwitz and Horvitz-Thompson estimators, and the *conservative* and *optimistic* stopping strategies on the performance of the framework.

		$RE \downarrow$			$loss_{er} \downarrow$		
		con1	con2	opt	con1	con2	opt
$recall_t = 1.0$	HH	0.003	-	0.023	0.159	-	0.144
	HT	0.000	0.000	0.000	0.170	0.170	0.170
$recall_t = 0.9$	HH	0.059	-	0.054	0.130.	-	0.120
	HT	0.060	0.063	0.073	0.108	0.132	0.097
$recall_t = 0.8$	HH	0.086	-	0.037	0.124	-	0.117
	HT	0.059	0.091	0.095	0.104	0.112	0.124

*: $\uparrow$ means that higher values are better, and $\downarrow$ means that lower values are better.

3.7 Conclusion

In this chapter, we have answered **RQ2** introduced in Chapter 1 by studying how to determine the stopping point of document selection in order to construct test collections that balance the cost of assessing document relevance and the gain of identifying relevant documents.

We propose a novel continuous active learning framework by jointly training a ranking model to rank documents, and conducting "greedy" sampling to estimate the total number of relevant documents in the collection. Within the framework we propose to use the AP-Prior as the sampling distribution, the Horvitz-Thompson or Hansen-Hurwitz estimator to estimate the total number of relevant documents, and the *optimistic* or *conservative* strategy to determine whether to stop the TAR process or not. We prove the unbiasedness of the proposed estimators under a with-replacement sampling design. To examine the effectiveness of the proposed framework, we compared it against the *Knee*, *Target*, *SCAL*, *SD-training* and *SD-sampling* methods and provided detailed analysis on various datasets including the CLEF Technology-Assisted Reviews in Empirical Medicine datasets [81, 83, 85], the TREC Total Recall datasets [60] and the TREC Legal datasets [41]. The experimental results demonstrate that the proposed method performs secondary to *Knee* but better than other baselines in terms of high recall and low cost; on the other hand, the proposed framework performs better than *Knee* and other baselines in terms of stopping the TAR process on time.

To sum up, the proposed framework combines the advantages of the continuous active learning approach with the advantages of sampling methods. It can effectively retrieve relevant documents similar to CAL but also provide a transparent, accurate, and effective stopping point. Specifically, it is recommended to use the AP-Prior distribution, the Horvitz-Thompson estimator together with the *conservative* stopping strategy. If efficiency is also considered, the Hansen-Hurwitz estimator is recommend instead

of the Horvitz-Thompson estimator.

There are limitations and several directions that we have not touched and can be followed in the future work. First, the sampling and estimation in the proposed framework is in a "sequential" manner, i.e., at each iteration, a certain number of documents are sampled and an estimator of R is calculated. When the sequence of estimators are repeatedly compared with the target recall until the desired result is observed – the estimated recall exceeds the target recall, the process yields a biased estimate of recall, even though the estimator at each iteration is unbiased. This issue is inherent for sequential testing [168]. Second, the proposed framework is designed for small-scale document collections. It requires a relatively small list of documents instead of a collection containing millions of documents so that the calculation of mean and variance of R is feasible. Currently, we adapted our framework on the large document collection of TR and LEGAL by randomly splitting the documents, running our algorithm, and then concatenating the sampled documents for the final reviewing. However, this is not the best solution. More work can be done such as training the ranking model globally to avoid sampling too many non-relevant documents. Third, like all other sampling methods, the proposed framework inherently has the high variance problem for low-prevalence topics. In practice, where prevalence is low, one must resort to solutions such as snowball sampling, capture-recapture, and other techniques that are common in biostatistics and medicine. It is worth studying how to integrate these solutions. Fourth, the performance of the ranking model can be further improved in order to produce a good ranked list of documents. Currently, only the document text instead of the topic text is employed for feature representation, the simple TF-IDF is used for document representation, the logistic regression model is used as the ranking model, and the ranking model is trained in a topic-wise manner, i.e., a new ranking model is trained from scratch for each topic. In future work, a stronger ranking model can be trained by addressing these issues. Finally, the current framework only considers the number of missing relevant documents to stop the TAR process. More factors should be taken into account such as the importance of the missing documents. For example, *risk of bias* and *quality*, two concepts from systematic review domain indicating how reliable the results of the studies included in a systematic review are, can be leveraged to determine stopping the TAR process [68].

In next chapter, we will study how to leverage crowdsourcing techniques to acquire relevance assessments of the selected documents and aggregate the noisy crowdsourcing assessments for test collection construction.

4

Aggregating Crowd Relevance Assessments

In Chapter 3, we have studied how to determine the stopping point of document selection for constructing test collections that balance the cost of assessing document relevance and the gain of identifying relevant documents. After the documents are selected, the next step is to acquire their relevance labels, which is often done through crowdsourcing. In this chapter, we investigate the issue of leveraging crowdsourcing techniques to acquire relevance assessments of the selected documents and aggregating the noisy crowdsourcing assessments for test collection construction. We aim to answer the following research question asked in Chapter 1:

RQ3 How can we effectively aggregate crowdsourcing labels in order to acquire high-quality labels in test collections?

4.1 Introduction

As we have explained in Chapter 1, test collections built through the Cranfield paradigm have significantly benefited the development of IR systems [32, 162]. They are used both as training data to develop new retrieval models and as test data to evaluate model performance. Building such test collections requires assessing the relevance of a large number of documents to a set of search queries. Traditionally the assessment of relevance is performed by trained professionals in a controlled lab environment [162]. As the need for creating new test collections to support the development of algorithms increases, so does the number of documents that need to be labeled. This makes collection construction expensive and time-consuming, if not financially unfeasible. Crowdsourcing has arisen as a natural cost-effective solution for the construction of test collections and has been widely adopted by the IR community [3, 62, 78, 88]. Typically, the task of obtaining the relevance label of a document to a query is assigned to crowd *workers* (or *annotators*) in the form of human intelligence tasks (HITs), which we call *tasks* for brevity.

Despite its cost-effectiveness, crowdsourcing introduces a major challenge – controlling the quality of the obtained labels through careful aggregation [92]. Majority voting has been the most prominent aggregation method, with early experiments showing that labels derived by a majority vote of multiple untrained crowd annotators can

reach a quality comparable to that of a trained NIST assessor [4]. At the same time, more recent work has shown that the quality of annotations is affected by a number of factors both from the annotator side and the task side, which are not considered by majority voting. The impact of human factors can be found in [88], which shows that an annotator's motivation, interest and familiarity with the task, perceived task difficulty, and satisfaction with the offered payment all influence the quality of the crowd annotations. The impact of task factors are studied in [63], concluding that task instruction, task subjectivity, task type, and the monetary reward of tasks all influence the quality of crowd annotations.

The aforementioned factors from the annotator side and the task side can be fused in two abstract concepts: the *difficulty* of the task and the *competence* of the annotator. Mainstream approaches to model the two factors for label aggregation are probabilistic graphical models (see the surveys of Li et al. [100] and Zheng et al. [182]). The hypothesis is that if a task is inherently easy, the labels from different crowd annotators will be consistent; otherwise, the labels will differ, and so modelling task difficulty may help improve the inference of the "true" label. Similarly, if an annotator is familiar with the topic of the search query and is well motivated (e.g., satisfied with the payment), he or she has more chance to give answers of good quality. The various probabilistic graphical models have made different assumptions of the annotation generation process where factors such as the difficulty of tasks and the competence of annotators are modelled. A detailed review of existing work can be found in Section 4.2.

The focus of this chapter lies on devising a novel label aggregation model that can handle noisy annotations. Our work is an extension of previous work in the sense that we assume a different annotation generation process. We use a Gaussian process (GP) for task correlation and multiple Gaussian distributions for the difficulty of tasks and the competence of annotators. The model allows us to integrate prior knowledge on tasks and model the label bias of tasks, the difficulty of tasks, the label bias of annotators and the competence of annotators.

More specifically, we propose a Bayesian generative model, the multi-annotator Gaussian process (MAGP) model, to tackle the problem of label aggregation. In particular, we make the hypothesis that *the true label of a document is correlated with the true labels of other documents close to it in some feature space*. This hypothesis has been examined extensively in IR community, e.g., in the *cluster hypothesis* [115]. To take the correlation across documents into consideration, we use a GP prior on the latent "true" labels of tasks. A very important consideration of using a GP prior is that we can infer the true labels of observed tasks, but also predict labels for future tasks; while existing methods using probabilistic graphical models are only able to infer the true labels of observed tasks. Furthermore, we assume that *the observed crowd label is a random variable, composed of the latent true label, noise from the task itself and noise from the annotator who gives the crowd label*. This is partially evidenced by Maddalena et al. [114] who have empirically shown that studied topic difficulty and relevance level (relevant or nonrelevant) affect crowd label quality. We use multiple Gaussian variables to model task noise and annotator noise, respectively. The advantage of such a model is that we can use the variance parameter of the Gaussian distribution to model how the labels vary, and use the mean parameter to model any bias towards or against relevancy, both per task and per annotator. In this chapter

we focus on the variance to model task difficulty and annotator competence and leave modelling bias as future work. Based on the two assumptions we propose a new annotation generation process: first, a Gaussian task noise and Gaussian annotator noise are added to the latent "true" label and then the final observed label is generated by passing the variable to a Bernoulli distribution. Based on the generative probabilistic model, we can calculate the likelihood of the observed crowd annotations. The latent "true" labels, the annotator competence, and the task difficulty are jointly learned through maximizing the log likelihood plus the log prior of these parameters.

The main contributions of this chapter are the following:

- We propose a new Bayesian generative model to capture the latent true labels, annotator competence, annotator's bias towards relevancy, task difficulty, and task's bias towards relevancy for observed tasks; further more, it can also predict labels for future tasks.
- We apply a variational expectation maximization (EM) algorithm to effectively learn the model parameters.
- We empirically demonstrate the effectiveness of our model on both synthetic data and real data in terms of inferring latent true labels and constructing high-quality test collections for IR evaluation.

The rest of the work is organized as follows: Section 4.2 gives a broader description of related work, Section 4.3 gives an overview of the Gaussian Process classification method, Section 4.4 provides a clear formulation of the problem and describes our contribution, Section 4.5 sets up the experiment, including a description of the datasets and the baselines, and Section 4.6 discusses the results. The chapter concludes with the final observations in Section 4.7.

4.2 Related Work

4.2.1 Crowdsourcing in Information Retrieval

The task of performing relevance assessment is critical to generate high quality IR evaluation collections. With the rising popularity of crowdsourcing relevance assessments, researchers have started to look at how to deal with the quality control challenge in crowdsourcing. TREC has organized crowdsourcing tracks from 2011 to 2013 [146–148], aiming to investigate the use of crowdsourcing techniques to evaluate information retrieval systems. Alonso et al. [3] have shown that the crowdsourcing assessments and the expert assessments produced by TREC assessors correlate well in IR evaluation measures.

The quality of crowd annotations are significantly affected by human factor. Han et al. [62] studied the impact of worker experience on fine-grained aspects such as working strategies, productivity levels, quality and diversity of the crowdsourcing annotations. Furthermore, researchers have investigated various quality control methods to ensure crowd annotation quality during the running of crowdsourcing tasks. For example, limiting the time available for relevance assessment was shown useful for an IR test collection of better quality [112], asking for the annotator's rationale such as a justification more than just a relevance assessment leads to more accurate relevance assessment [118].

Existing label aggregation models have been applied to acquire relevance labels for IR test collection. Hosseini et al. [73] applied the Dawid and Skene model [46] and found that it is better than the majority voting method. Models specifically designed to infer latent true relevance labels have been studied. Davtyan et al. [45] exploited document content to infer the true relevance label from crowd labels; they proposed label aggregation algorithms by utilizing majority voting with nearest neighbor or Gaussian process classification. Ferrante et al. [54] proposed an interesting model where they consider relevance assessment as a stochastic process and propose a novel label aggregation model.

The focus of this chapter is on the label aggregation task of IR crowdsourcing. For other works except label aggregation the reader is referred to [2].

4.2.2 Probabilistic Models for Aggregating Crowd Annotations

The problem of learning from multiple noisy annotations has attracted research attention since crowdsourcing became popular. Li et al. [100] and Zheng et al. [182] reviewed existing work on crowdsourcing label aggregation. The goal of label aggregation is to infer the true label of each task given redundant and maybe inconsistent crowd annotations. The mainstream solution is to design a probabilistic graphical model to model the probability that an annotator produces a specific label (e.g., relevant) for a task. Usually each observed label is assumed to be generated independently from other observed labels, given the latent true label of the task.

Let y_i^j denote the annotator j's label to task i, and z_i the latent true label for task i, and the corresponding capital characters as the variable for all tasks or annotators. The mainstream probabilistic graphical models define the joint distribution of observed labels $\boldsymbol{Y}$ and $\boldsymbol{z}$ as $p\left(\boldsymbol{Y}, \boldsymbol{z} \mid \boldsymbol{\theta}\right) = \prod_i p\left(z_i \mid \boldsymbol{\theta}\right) \prod_j p\left(y_i^j \mid z_i, \boldsymbol{\theta}\right)$, but the major difference is the way they model $p\left(z_i \mid \boldsymbol{\theta}\right)$ and $p\left(y_i^j \mid z_i, \boldsymbol{\theta}\right)$. The parameter $\boldsymbol{\theta}$ in the model involves factors like the difficulty of each task and the competence of each annotator etc. The likelihood of the observed annotations is therefore $p\left(\boldsymbol{Y} \mid \boldsymbol{\theta}\right) = \int_{\boldsymbol{z}} p\left(\boldsymbol{Y}, \boldsymbol{z} \mid \boldsymbol{\theta}\right) \mathrm{d}\boldsymbol{z}$. The parameter values $\boldsymbol{\theta}$ are then inferred by maximizing the likelihood of the observed data. Representative methods include the Dawid and Skene (DS) model [46], the learning from crowd (LFC) model, which is a Bayesian version of DS [131], the independent Bayesian classifier combination (iBCC) model [91], the generative model of labels, abilities, and difficulties (GLAD) [171], the DARE model [14], and the MACE model [74] etc.

DS [46] models $p\left(y_i^j \mid z_i\right)$ by a confusion matrix $v_{jkl} = p\left(y_i^j = l \mid z_i = k\right)$ for each annotator j, which can be understood as an annotator competence matrix, and models $p\left(z_i\right)$ by a categorical distribution $\tau_k = p\left(z_i = k\right)$; then the EM algorithm is employed to optimize model parameters v_{jkl} and τ_k. LFC [131] extends DS by adding a Dirichlet prior for $\boldsymbol{v_{jk}}$ and $\boldsymbol{\tau}$; again, EM is employed to optimize model parameters. iBCC [103] is a Bayesian version of LFC; Gibbs sampling is used for parameter inference. GLAD [171] models $p\left(y_i^j \mid z_i\right)$ by a logistic function $\frac{1}{1+e^{-\alpha_i \beta_j}}$ where α_i is the difficulty of task i and β_j is the competence of annotator j, and models $p\left(z_i\right)$ by the same categorical distribution $\tau_k = p\left(z_i = k\right)$; model parameters are largely compressed into $M + N$ instead of $M \times K \times K$ as in DS; the parameters are inferred with the EM algorithm. DARE [14] models $p\left(y_i^j \mid z_i\right)$ by a mixture of two distributions

conditioned on whether the annotator knows the answer; if so, the model constrains y_i^j to match the true label z_i, otherwise it assumes y_i^j is uniformly sampled from the available classes; whether the annotator knows the answer depends on the task difficulties and the annotator competences; inference employs the expectation propagation (EP) algorithm. MACE [74] models $p\left(y_i^j = l \mid z_i = k\right)$ using a confusion matrix in a similar way as DS, where $v_{jkl} = (1 - \theta_j)\epsilon_{jl}$ if $k \neq l$ and $v_{jkl} = \theta_j + (1 - \theta_j)\epsilon_{jl}$ if $k = l$; again, this confusion matrix reduces the number of parameters; inference uses the EM algorithm. As these methods assume different annotation generation processes, it is difficult to determine which performs the best in practice [102].

Another line of work jointly tackles label aggregation problems with other tasks such as downstream classification task and evaluation of retrieval systems. For example, Zhan et al. [180] proposed a noise-aware classification framework that integrates both noisy label aggregation and classification; Ferrante et al. [55] addressed the problem of aggregating crowd annotations from a new perspective where they model crowd relevance judgements as sources of uncertainty and design IR evaluation metrics such as average precision based on the crowd annotations.

4.2.3 Gaussian Process for Aggregating Crowd Annotations

A Gaussian process is a stochastic process with the important characteristics that any finite number of random variables follow a joint Gaussian distribution [5]. Mathematically, it is equivalent to many models such as Bayesian linear models, spline models, and large neural networks. A formal definition is given: a GP is a collection of random variables $\{f_i \mid i \in I\}$ satisfying that any finite set of $\{f_i\}$ follows a multivariate Gaussian distribution [128], where I is an infinite index set. A GP is completely specified by its mean function $m(\boldsymbol{x})$ and covariance function $k\left(\boldsymbol{x}, \boldsymbol{x}'\right)$. We say $\boldsymbol{f}$ is a Gaussian process and write it as $\boldsymbol{f} \sim \mathcal{GP}\left(m\left(\boldsymbol{x}\right), k\left(\boldsymbol{x}, \boldsymbol{x}'\right)\right)$. The mean function $m\left(\cdot\right)$ describes the average values of the infinite latent variables we expect without seeing any training data. It allows one to incorporate prior knowledge about the latent variables we wish to model. For example, any kind of mechanical or physical model can be used as the mean function. The covariance function $k\left(\cdot, \cdot\right)$ describes the correlation between two latent variables. Two latent variables are strongly correlated only if the corresponding inputs are close to each other.

There are several benefits to using a GP for label aggregation. First, it allows us to incorporate auxiliary information of tasks by assuming a GP prior over tasks. For example, in our case, we incorporate the textual information of query-document pairs; while most existing probabilistic graphical models are only able to incorporate crowd labels. Second, a GP model works especially well for small data [128]. Small number of tasks budgeted for constructing datasets is common in many crowdsourcing scenarios [78], and a GP model fits it better than data-hunger neural networks. Finally, a GP naturally quantifies the uncertainty of a prediction which can help to determine whether to include a task in the dataset to be constructed. Whereas the aforementioned probabilistic graphical models do not have these benefits.

Several GP-based models have been proposed for label aggregation task [59, 120, 134, 137]. Groot et al. [59] studied the problem of inferring the true real-valued label given multiple noisy real-valued labels. By averaging multiple crowd labels, each weighted by the variance of the corresponding annotator, to one single label, the ori-

ginal task is reduced to a vanilla Gaussian process regression task. Rodrigues et al. [134] proposed a GP-based model to account for multiple annotators with different levels of expertise. The data generation process they assume is that for each input point there is a latent true label, and importantly a GP is assumed the prior of all the latent true labels; then the corresponding annotator produces a label through two Bernoulli distributions. They use EP to optimize model parameters. Ruiz et al. [137] proposed a model of binary label aggregation by adding a GP prior on top of the confusion matrix in the DS model. A novel variational inference algorithm is proposed for the model. Further, the model is extended in order to deal with large-scale datasets, e.g., with approximately 1 million tasks, in the work of [120]. Our work is different from these models in the sense that it assumes a different annotation generation process, and accordingly, a different inference method is used to learn model parameters.

4.3 Gaussian Process Classification

In this section we provide a description of the Gaussian process classification model. Given a set of training samples $\boldsymbol{C} \triangleq \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N} \triangleq (\boldsymbol{X}, \boldsymbol{y})$, where N is the number of samples in C, $\boldsymbol{x}_i$ is the input point and y_i is the corresponding class label, we want to predict class probabilities for a new point $\boldsymbol{x}_*$.

A Gaussian process classification model assumes the observed data are generated through the following process: a Gaussian process first maps the input point $\boldsymbol{x} \in \mathbb{R}^D$ to a latent variable $f \in \mathbb{R}$, then a link function maps f to a real value $y \in [0, 1]$. The link function can be a *logistic* function or a *probit* function. We use a probit function $\Phi(f) \triangleq \int_{-\infty}^{f} \mathcal{N}(z \mid 0, 1)\, \mathrm{d}z$ in this chapter. In binary classification setting, we denote the positive class as 1 and negative class as 0. Therefore, according to the property of $\Phi(f)$, the positive class probability is $p(y = 1 \mid \boldsymbol{x}) = \Phi(f)$ and the negative class probability is $p(y = 0 \mid \boldsymbol{x}) = \Phi(-f)$.

The data generation process can be rephrased in a perspective of random variables. First, the latent variables $\boldsymbol{f}$ follow a Gaussian process, denoted by:

$$\boldsymbol{f} \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')) \,. \tag{4.1}$$

Second, each observed variable y follows a Bernoulli distribution conditioned on its corresponding latent variable f, denoted by:

$$y \mid f \sim Bernoulli(\Phi(f)) \,. \tag{4.2}$$

Now let us calculate the predictive probability of a new point $\boldsymbol{x}_*$ being classified as positive $p(y_* = 1 \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})$. We need to first compute the posterior distribution of the latent variable f_* when the training data $\boldsymbol{C}$ being observed using Equation (4.3),

$$p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{f})\, p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y})\, \mathrm{d}\boldsymbol{f} \,, \tag{4.3}$$

and then compute the positive class probability using

$$p(y_* = 1 \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \Phi\left(\int f_* p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})\, \mathrm{d}f_*\right) = \Phi(\mathbb{E}[f_*]) \,. \tag{4.4}$$

Equation (4.4) is straightforward to calculate if we know $p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})$. Next, we introduce how to calculate $p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{f})$ and $p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y})$ in Equation (4.3). Given the training inputs and the new input point $[\boldsymbol{X}, \boldsymbol{x}_*]$, the corresponding latent variable $[\boldsymbol{f}, f_*]$ follows a multivariate Gaussian distribution

$$\begin{bmatrix} \boldsymbol{f} \\ f_* \end{bmatrix} \Bigg| \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{x}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \mu_* \end{bmatrix}, \begin{bmatrix} K & K_* \\ K_*^T & K_{**} \end{bmatrix} \right), \tag{4.5}$$

where $\boldsymbol{\mu} \triangleq m(\boldsymbol{X})$ is the mean vector of X, $\mu_* \triangleq m(x_*)$ is the mean value at x_*. $K \triangleq \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$ is the covariance matrix of X, $K_*^T \triangleq [k(\boldsymbol{x}_1, \boldsymbol{x}_*), \ldots, k(\boldsymbol{x}_N, \boldsymbol{x}_*)]$ is the covariance vector between X and $\boldsymbol{x}_*$, and finally $K_{**} \triangleq k(\boldsymbol{x}_*, \boldsymbol{x}_*)$ is the covariance between $\boldsymbol{x}_*$ and $\boldsymbol{x}_*$.

Based on the conditional Gaussian distribution rule [128], the distribution of f_* conditioned on $\boldsymbol{f}$ is

$$f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{f} \sim \mathcal{N}(\boldsymbol{\mu}_{f_*}, \boldsymbol{\Sigma}_{f_*}), \tag{4.6}$$

where $\boldsymbol{\mu}_{f_*} = m(\boldsymbol{x}_*) + K_* K^{-1}(\boldsymbol{f} - m(\boldsymbol{X}))$, $\boldsymbol{\Sigma}_{f_*} = K_{**} - K_*^T K^{-1} K_*$. Now let us approximate $p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y})$, the joint posterior distribution of the latent variables. By using Bayes rule, $p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y})$ becomes

$$p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{f} \mid \boldsymbol{X})\, p(\boldsymbol{y} \mid \boldsymbol{f})}{p(\boldsymbol{y} \mid \boldsymbol{X})}. \tag{4.7}$$

The prior $p(\boldsymbol{f} \mid \boldsymbol{X})$ is presented in Equation (4.5), and the likelihood $p(\boldsymbol{y} \mid \boldsymbol{f})$ is

$$p(\boldsymbol{y} \mid \boldsymbol{f}) = \prod_{i=1}^{N} p(y_i \mid f_i) = \prod_{i=1}^{N} \Phi\left((-1)^{(1-y_i)} f_i \right). \tag{4.8}$$

However, the computation of the evidence $p(\boldsymbol{y} \mid \boldsymbol{X})$ is not trivial, because it is not a Gaussian distribution due to the multiplication of the Bernoulli likelihood with the Gaussian prior. The solution can be either analytic approximations of integrals like Expectation Propagation or Laplace Approximation, where a Gaussian distribution is used to approximate the posterior distribution $p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y})$ and the evidence $p(\boldsymbol{y} \mid \boldsymbol{X})$, or numerical approximation methods like Monte Carlo sampling [128].

In general, the mean function $m(\cdot)$ and the covariance function $k(\cdot, \cdot)$ of the Gaussian process classification model both contains parameters to be learned. To learn these parameters, one needs to maximize the likelihood of the observed data, which is $\arg\max_{\boldsymbol{\theta}} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$; or adopt a Bayesian view by considering both the likelihood and the prior of parameters, which is $\arg\max_{\boldsymbol{\theta}} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$. [1] The prior item $p(\boldsymbol{\theta})$ can avoid model overfitting and the likelihood $p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$ "incorporates a trade-off between model fit and model complexity" [128].

[1] Here we rewrite $p(\boldsymbol{y} \mid \boldsymbol{X})$ as $p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$ when we explicitly take $\boldsymbol{\theta}$ into consideration.

4.4 Multi-Annotator Gaussian Process Classification

4.4.1 Problem Formulation

Different from the vanilla Gaussian process classification model where in the observed date each example has one single class label, in the multi-annotator case, each example (or *task*) are given multiple class labels annotated by different annotators. Assume there are N unique tasks and M unique annotators in training set. Denote $\boldsymbol{C}_i \triangleq \{(\boldsymbol{x}_i, y_i^1), (\boldsymbol{x}_i, y_i^1), ..., (\boldsymbol{x}_i, y_i^{M_i})\}$ the annotated labels for the i-th task. Hence, the observed data is denoted by $\boldsymbol{C} \triangleq \{\boldsymbol{C}_1, \boldsymbol{C}_2, ..., \boldsymbol{C}_N\}$. Our goal is to infer the true class label y_i for each task $\boldsymbol{x}_i$.

4.4.2 The Model

The Assumptions

As discussed in the introduction, we assume that the observed annotation labels are affected by three major factors, the latent true label of each task, the difficulty level of each task, and the competence of each annotator.

First, to a big extent it is the the latent true label that determines the observed label. Further, true labels across topically similar documents should be correlated. To this end, we use a Gaussian process $\boldsymbol{f} \triangleq [f_1, f_2, ..., f_N]$ to model the latent true labels,

$$\boldsymbol{f} \sim \mathcal{GP}\left(m\left(\boldsymbol{x}\right), k\left(\boldsymbol{x}, \boldsymbol{x}'\right)\right) , \tag{4.9}$$

where the covariance function $k\left(\boldsymbol{x}, \boldsymbol{x}'\right)$ captures the correlation across query-document pairs, and the mean function $m\left(\boldsymbol{x}\right)$ captures our prior knowledge on the relevance of query-document pairs.

Second, the difficulty level of a task affects the consistency of its annotated labels. If a task is easy the annotations from different crowd annotators will tend to be the same; otherwise, the annotations will be very different. To this end, we use a Gaussian noise ϵ_i to model task difficulty for each task $\boldsymbol{x}_i$:

$$\epsilon_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right) , \tag{4.10}$$

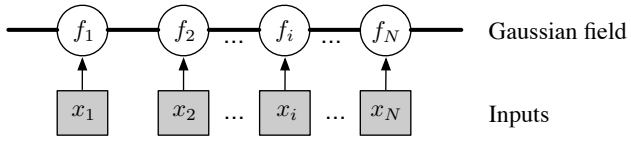
where μ_i models the bias towards relevancy or irrelevancy and σ_i^2 models the difficulty level of the task.

Third, the competence of an annotator affects the quality of his or her annotations. Note that by "competence" we mean a number of factors that can affect an annotator's quality for a given task, including his ability, familiarity with the task, satisfaction from the payment, or even the annotator's preference or special ability towards some specific topics.

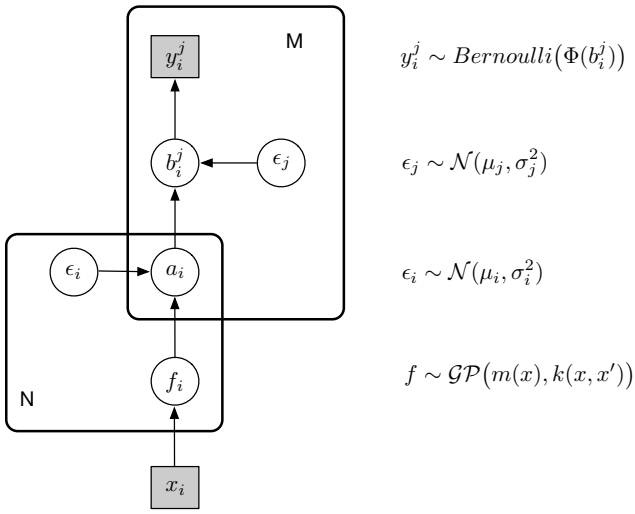
To model all these, similarly to ϵ_i, we use another Gaussian noise ϵ^j:

$$\epsilon^j \sim \mathcal{N}\left(\mu_j, \sigma_j^2\right) , \tag{4.11}$$

where the mean μ_j models the bias of an annotator towards relevancy or irrelevancy, and the variance σ_j^2 models his or her competence.



(a) Prior



(b) Likelihood

Figure 4.1: Graphical model for the Multi-Annotator Gaussian process model. Squares represent observed variables and circles represent latent variables. Figure 4.1(a) illustrates the prior of the annotation process. The thick horizontal bar represents a set of fully connected nodes $\{f_i\}$ which follows a Gaussian process. Figure 4.1(a) illustrates the likelihood. For each annotator a latent variable is generated via $a_i^j = f_i + \epsilon_i$ and latent variable is generated via $b_i^j = a_i^j + \epsilon_j$, and finally, the observed variable is generated via a Bernoulli distribution $Bernoulli\left(\Phi\left(b_i^j\right)\right)$.

Annotation process

We assume the observed data are generated through the following process, which is illustrated as a chain graphical model in the bottom panel of Figure 4.1. Each task $\boldsymbol{x}_i \in \mathbb{R}^D$ is associated with a latent variable $f_i \in \mathbb{R}$, which indicates its relevance label. These latent variables conform to a Gaussian process. When a task $\boldsymbol{x}_i$ is distributed to an annotator A_j, a Gaussian noise ϵ_i is added to f_i to generate a_i^j which incorporates the difficulty of the task, and then another Gaussian noises ϵ^j is added to a_i^j to generate the latent variable b_i^j which incorporates the competence of the annotator. We further assume that ϵ_i is independent from f_i and any other noise $\epsilon_{l(l\neq i)}$; and ϵ^j is independent from any f_i, any ϵ_i, and any other $\epsilon^{k(k\neq j)}$. Finally a probit function maps b_i^j to a value in the interval [0, 1], denoted by $\Phi\left(b_i^j\right)$, which represents the probability of y_i^j being relevant, i.e. $p\left(y_i^j = 1\right) = \Phi\left(b_i^j\right)$.

Label Inference

Given a new point or an existing point $\boldsymbol{x}_*$, our goal is to predict the class label. Similar to Equation (4.4) of the vanilla Gaussian process classification model, we calculate the positive class probability using $p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{Y})$ in Equation (4.12):

$$p(y_* = 1 \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{Y}) = \Phi\left(\int f_* p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{Y}) \, \mathrm{d}f_*\right) = \Phi\left(\mathbb{E}[f_*]\right) . \quad (4.12)$$

The integral item $p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{Y})$ is further represented as:

$$p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{Y}) = \int p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{f}) \, p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{Y}) \, \mathrm{d}\boldsymbol{f} . \quad (4.13)$$

Similar to the vanilla Gaussian process classification model, using the Bayes' rule $p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{Y})$ in Equation (4.13) can be rewritten as:

$$p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{Y}) = \frac{p(\boldsymbol{f} \mid \boldsymbol{X}) \, p(\boldsymbol{Y} \mid \boldsymbol{f})}{\int p(\boldsymbol{f} \mid \boldsymbol{X}) \, p(\boldsymbol{Y} \mid \boldsymbol{f}) \, \mathrm{d}\boldsymbol{f}} = \frac{p(\boldsymbol{f} \mid \boldsymbol{X}) \, p(\boldsymbol{Y} \mid \boldsymbol{f})}{p(\boldsymbol{Y} \mid \boldsymbol{X})} . \quad (4.14)$$

The prior $p(\boldsymbol{f} \mid \boldsymbol{X})$ is the Gaussian prior, presented in Equation (4.5). The likelihood of the observed data from multiple annotators, $p(\boldsymbol{Y} \mid \boldsymbol{f})$, consists of multiple Bernoulli likelihood. It is one of the contribution of this work. Based on the independence assumption in the annotation generation process, $p(\boldsymbol{Y} \mid \boldsymbol{f})$ can be presented as:

$$p(\boldsymbol{Y} \mid \boldsymbol{f}) = \prod_{i=1}^{N} \prod_{j=1}^{M} \int \mathcal{N}\left(b_i^j \mid f_i + \mu_i + \mu_j, \sigma_i^2 + \sigma_j^2\right) \Phi\left((-1)^{(1-y_i^j)} b_i^j\right) \mathrm{d}b_i^j \quad (4.15)$$

Below we give the detailed derivation.

$$p(\boldsymbol{Y} \mid \boldsymbol{f}) = \prod_{i=1}^{N} \prod_{i=1}^{M} p\left(y_i^j \mid f_i\right) \quad (4.16a)$$

$$p\left(y_i^j \mid f_i\right) = \iint p\left(y_i^j \mid a_i^j, b_i^j, f_i\right) p\left(a_i^j, b_i^j | f_i\right) \mathrm{d}a_i^j \, \mathrm{d}b_i^j \quad (4.16b)$$

$$= \iint p\left(a_i^j \mid f_i\right) p\left(b_i^j \mid a_i^j\right) p\left(y_i^j \mid b_i^j\right) \mathrm{d}a_i^j \, \mathrm{d}b_i^j \quad (4.16c)$$

$$= \iint \mathcal{N}\left(a_i^j \mid f_i + \mu_i, \sigma_i^2\right) \mathcal{N}\left(b_i^j \mid a_i^j + \mu_j, \sigma_j^2\right) \Phi\left((-1)^{1-y_i^j} b_i^j\right) \mathrm{d}a_i^j \, \mathrm{d}b_i^j \quad (4.16d)$$

$$= \int \mathcal{N}\left(b_i^j \mid f_i + \mu_i + \mu_j, \sigma_i^2 + \sigma_j^2\right) \Phi\left((-1)^{(1-y_i^j)} b_i^j\right) \mathrm{d}b_i^j \quad (4.16e)$$

Equation (4.16b) applies the Total Probability rule; Equation (4.16c) is true because a_i^j is only dependent on f_i, b_i^j is only dependent on a_i^j, and y_i^j is only dependent on b_i^j; Equation (4.16d) is true because the sum of a constant f_i and a Gaussian variable ϵ_i is a Gaussian variable, similarly the sum of a constant a_i^j and a Gaussian variable

ϵ_j is a Gaussian variable; Equation (4.16e) integrates a_i^j out by applying the Gaussian Marginal and Conditional rule [121, page 93].

Now we continue the discussion of Equation (4.14). Note that $p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{Y})$ is not a Gaussian distribution due to the multiplication of the Bernoulli likelihood with the Gaussian prior and thus the computation of the integral is not trivial. The major idea is to use a multivariate Gaussian distribution $q(\boldsymbol{f})$ to approximate $p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{Y})$. The problem is solved together with the optimization of model parameters.

4.4.3 Model Optimization

The proposed model contains the parameters of the mean function and the covariance function which are common for all Gaussian process models, the parameters (μ_i, σ_i^2) for every task i and (μ_j, σ_j^2) for every annotator j.

Our goal is to optimize the model with regard to $\boldsymbol{\theta}$. Similar to the vanilla Gaussian process classification model, we adopt a Bayesian view and maximize log of the likelihood of the observed data plus the log of the parameter prior: $\log p(\boldsymbol{Y}, \boldsymbol{\theta} \mid \boldsymbol{X}) = \log p(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$ with regard to $\boldsymbol{\theta}$. We formally write the problem as:

$$\arg\max_{\boldsymbol{\theta}} \left\{ \log p(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \right\} . \tag{4.17}$$

As the first part $\log p(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = \int p(\boldsymbol{f} \mid \boldsymbol{X}) p(\boldsymbol{Y} \mid \boldsymbol{f}) \, \mathrm{d}\boldsymbol{f}$ is intractable, which is explained in the discussion of Equation (4.14), we instead maximize its *variational lower bound* (also called *evidence lower bound*, ELBO), which is tractable. The derivation of ELBO is presented as follows:

$$\begin{aligned}
\log p(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{\theta}) &\triangleq \log p(\boldsymbol{Y}) = \int q(\boldsymbol{f}) \log p(\boldsymbol{Y}) \, \mathrm{d}\boldsymbol{f} \\
&= \int q(\boldsymbol{f}) \log \frac{p(\boldsymbol{Y}, \boldsymbol{f})}{q(\boldsymbol{f})} \, \mathrm{d}\boldsymbol{f} + \left(- \int q(\boldsymbol{f}) \log \frac{p(\boldsymbol{f} \mid \boldsymbol{Y})}{q(\boldsymbol{f})} \, \mathrm{d}\boldsymbol{f} \right) \\
&= \int q(\boldsymbol{f}) \log \frac{p(\boldsymbol{Y}, \boldsymbol{f})}{q(\boldsymbol{f})} \, \mathrm{d}\boldsymbol{f} + KL\left(q(\boldsymbol{f}) || p(\boldsymbol{f} \mid \boldsymbol{Y})\right) \\
&\geqslant \int q(\boldsymbol{f}) \log \frac{p(\boldsymbol{Y}, \boldsymbol{f})}{q(\boldsymbol{f})} \, \mathrm{d}\boldsymbol{f} \\
&= \mathbb{E}_{q(\boldsymbol{f})} \left[\log p(\boldsymbol{Y} \mid \boldsymbol{f}) \right] - KL[q(\boldsymbol{f}) || p(\boldsymbol{f})] \triangleq ELBO ,
\end{aligned} \tag{4.18}$$

where $q(\boldsymbol{f}) \triangleq q(\boldsymbol{f} \mid \boldsymbol{\psi})$ is the parameterized variational functional to be learned, which is assumed a multivariate Gaussian distribution approximating $p(\boldsymbol{f} \mid \boldsymbol{Y})$; and $p(\boldsymbol{f}) \triangleq p(\boldsymbol{f} | \boldsymbol{\theta})$ is the prior distribution of the latent variables in Equation (4.5) and $p(\boldsymbol{Y} \mid \boldsymbol{f}) \triangleq p(\boldsymbol{Y} \mid \boldsymbol{f}, \boldsymbol{\theta})$ is the likelihood of the observed data in Equation (4.15), they are both parameterized by $\boldsymbol{\theta}$. Therefore we also denote $ELBO \triangleq ELBO(\boldsymbol{\psi}, \boldsymbol{\theta})$.

Finally, we adopt the variational Gaussian approximation method [125] and EM algorithm [122] to maximize Equation (4.17), which is reduced to $ELBO(\boldsymbol{\psi}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$. Both the E and M step are seen as maximizing the same function. The E step maximizes it with respect to the parameters of $q(\boldsymbol{f})$. Note that the $q(\boldsymbol{f})$ obtained in the E step is an approximation of the posterior distribution $p(\boldsymbol{f} \mid \boldsymbol{Y})$ according to Opper et al. [125]. The M step maximizes it with respect to the model parameters $\boldsymbol{\theta}$. The entire method is summarized in Algorithm 9.

Algorithm 9: Optimization of the MAGP model.

Input: A set of tasks and the corresponding crowdsourcing annotations $(\boldsymbol{X}, \boldsymbol{Y})$.
Output: A set of true task labels $\boldsymbol{y}$.

Initialize $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$;
while not converge **do**
  E step Fix $\boldsymbol{\theta}^{old}$ and maximize $ELBO\left(\boldsymbol{\psi}, \boldsymbol{\theta}^{old}\right) + \log p\left(\boldsymbol{\theta}^{old}\right)$ using the variational Gaussian approximation method [125] to get new $\boldsymbol{\psi}^{new}$;
  M step Fix $\boldsymbol{\psi}^{new}$ and maximize $ELBO\left(\boldsymbol{\psi}^{new}, \boldsymbol{\theta}\right) + \log p\left(\boldsymbol{\theta}\right)$ with respect to $\boldsymbol{\theta}$ using gradient descent to get $\boldsymbol{\theta}^{new}$;
  Check for convergence the relative error between the last and the current objective functions of Equation (4.17);
end while
Use optimal $\boldsymbol{\theta}^*$ and $\boldsymbol{\psi}$ to predict true label $\boldsymbol{y}$ by calculating Equation (4.12);

4.4.4 Task Representation

Different from typical label aggregation models using probabilistic graphical models (e.g., DS [46], LFC [131], iBCC [103], GLAD [171], DARE [14], and MACE [74]), where tasks are represented as a set of indicators of $\{1, 2, \ldots, N\}$, the covariance function $k\left(\boldsymbol{x}, \boldsymbol{x}'\right)$ in our model requires the tasks – query-document pairs – to be represented as a set of vectors of $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$. The vectors can incorporate any auxiliary information of query-document pairs. In this work, in order to capture multiple types of auxiliary information, the vectors are composed of lexical features, semantic features and ranking features. The three types of features have been demonstrated effective in relevance prediction tasks. Besides, the three types of features are purely based on the texts of queries and documents, which are available for most retrieval tasks, making our model easy to apply.

Lexical features We consider several lexical features which have been demonstrated effective in learning to rank algorithms [108]: (1) a term frequency score measuring the number of times that each term of a query occurs in a document (denoted by $\sum_{t \in q \cap d} TF(t, d)$), (2) a inverse document frequency score measuring how much information each term of a query provides ($\sum_{t \in q} IDF(t)$), (3) a TF-IDF score measuring both (denoted by $\sum_{t \in q \cap d} TF(t, d)\, IDF(t)$), (4) a cosine value between TF-IDF vectors of a query and a document, measuring lexical similarity between a query and a document, (5) a BM25 score between a query and a document, measuring lexical similarity between a query and a document ($\sum_{t \in q} IDF(t)\, \frac{TF(t,d)\,(k_1+1)}{TF(t,d)+k_1\,(1-b+b\,\frac{|d|}{avgdl})}$), and (6) a probability of observing a query given a document, which is based on language modelling method and measures lexical similarity between a query and a document [178] (denoted by $\sum_{t \in q \cap d} \log \frac{p\,(t|d)}{\alpha_d\, p\,(t|C)} + |q| \log \alpha_d + \sum_t \log p\,(t \mid C)$). In the aforementioned formulas, q is the query, d is the document, $|q|$ is the number of terms in q, $|d|$ is the number of terms in d, $avgdl$ is the average document length in a document collection, C is the document collection, k_1, b, and α_d are hyperparameters.

Semantic features Semantic features are important supplementary of lexical features in terms of capturing the correlation between query-document pairs. We fine-tune a

pre-trained BERT model [49] on the relevance classification datasets of the TREC Web 2009, 2010 and 2011 tracks [29, 30, 150], and then take each query-document pair as the input and output vector as it semantic representation. Fine-tuning a pre-trained BERT model on downstream tasks like relevance classification have been widely proven effective [164]. The three web tracks are used as the fine-tuning dataset because they share the same ClueWeb09 document collection with the crowdsourcing datasets in our experiments in Section 4.5.

Ranking features Except for features derived from textual information of queries and documents, we also use ranking features which are the ranks of documents to queries. The ranks are obtained from available ranked lists of retrieval systems. Combining multiple ranked lists produced by various retrieval systems (known as *meta search*) does help relevance classification [8]. In this work, we use all the available ranked lists, 35 in total, produced by the participating teams in the TREC 2009 million query track. We use the TREC 2009 million query track because the queries and documents of it cover that of the crowdsourcing datasets we are using in our experiments in Section 4.5.

4.4.5 Mean and Covariance Function

The mean function of the Gaussian process prior describes the average values of the latent variables $\boldsymbol{f}$ we expect without seeing any training data. It allows to incorporate any prior knowledge about document relevance to a query. To fully utilize this merit of the proposed model, we can set the mean function as any relevance classification model pre-trained on existing training data.

A relevance classification model can be understood in a function view. An input point $\boldsymbol{x}$ is first mapped to a real value z using Equation (4.19), and then z is mapped to a real value $o \in [0, 1]$ (i.e. the relevance probability) using Equation (4.20).

$$z = g(\boldsymbol{x}) \tag{4.19}$$

$$o = l(\boldsymbol{z}) \ . \tag{4.20}$$

For example, the mapping $g(\cdot)$ can be a feedforward neural network and the link function can be the *sigmoid* function. If the relevance classification model is trained on a training dataset, function $g(\cdot)$ can learn the prior knowledge of relevance. Consequently, we can use $g(\cdot)$ as the mean function to obtain this prior knowledge of relevance.

In our case, we use the pre-trained BERT transformer plus a linear layer [49] as $g(\cdot)$, and the *sigmoid* function as the link function. We train the relevance classification model on the datasets in the TREC Web 2009, 2010 and 2011 tracks, as they share the same ClueWeb09 document collection with the crowdsourcing dataset.

The covariance function describes the correlation between two latent variables. Two latent variables are strongly correlated only if the corresponding inputs are close to each other. For most covariance functions, there are two parameters: the length scale and signal variance. The length scale indicates with what scale the changes of input will not cause "large" change in output. The signal variance indicates the amplitude of the functions we model. We employ the linear covariance denoted by Equation (4.21) for semantic features and the RBF covariance denoted by Equation (4.22) for lexical

and ranking features.

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \boldsymbol{x} \cdot \boldsymbol{x}' \tag{4.21}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{l^2}\right) \tag{4.22}$$

4.4.6 Implementation

We employ GPflow [117] to implement our model. Specifically, we implement a new likelihood class named `GaussianBernoulliMix` for the calculation of Equation (4.15); we use `VGP_opper_archambeau` for the calculation of the right part of Equation (4.18). The model contains a number of parameters including the length scale and variance of the covariance function, the mean and variance of the Gaussian noise for the annotators, the mean and variance of the Gaussian noise for the tasks. We fix all the mean parameters as zero. We set $Gamma(1, 1)$ as the prior distribution of the remaining parameters. The other settings are the same as the default GPflow. The code is publicly available. [2]

4.5 Experimental Setup

4.5.1 Research Questions

In the remainder of the work we aim to answer the following research questions:
RQ1 Is the model able to learn the true labels from crowd labels, and how is the performance compared with baselines?
RQ2 Is the model able to correctly predict labels for new tasks?
RQ3 What is the effect of the annotation quality and annotation redundancy on model performance?
RQ4 What is the effect of different features in task representation on model performance?

4.5.2 Dataset

Real data

We test our label aggregation method on two crowdsourcing datasets: the crowdsourcing dataset of the TREC 2010 relevance feedback track [24] (CS2010[3]), and the crowdsourcing development dataset of TREC 2011 crowdsourcing track aggregation task (CS2011[4]). Each example in the two crowdsourcing datasets contains information of query ID, document ID, annotator ID, ground truth label, and crowd label. Remember that we need the axillary information of each query-document pair to get its vector representation, thus we extract query text from the corresponding topic file, we extract document text from the ClueWeb09 corpus, we extract document ranks to queries from rankings submitted by participation retrieval systems in the TREC 2009 million query track because the queries and documents cover that of the two crowdsourcing datasets.

[2] `https://github.com/dli1/magp`
[3] `https://www.ischool.utexas.edu/~ml/data/trec-rf10-crowd.tgz`
[4] `https://sites.google.com/site/treccrowd/2011`

The original CS2010 dataset contains 100 queries and 19,902 documents selected from the TREC 2009 million query track, and in total 20,232 unique query-document pairs and 96,883 relevance annotations given by 766 annotators. The corresponding ground truth label for each query-document pair is generated from NIST experts as previous TREC tracks. Among the crowd annotations we remove those that are marked as invalid due to reasons such as broken links, or that have no corresponding ground truth label, or that have no text or ranks available for the document. Consequently, there remains 3,275 unique query-document pairs and 18,479 relevance annotations with ground truth labels. The original dataset is judged on a ternary scale: highly relevant, relevant, and non-relevant. As our model is designed for binary labels, we turn the ternary scale into a binary scale by mapping highly relevant or relevant labels to relevant labels. The original CS2011 dataset contains 25 queries and 3,557 documents from the TREC 2009 million query track, and in total 3,568 unique query-document pairs and 10,752 binary relevance annotations given by 181 annotators. The corresponding ground truth labels are also from NIST experts. Similarly, invalid annotations are removed, resulting in 711 unique query-document pairs and 2,181 relevance annotations with ground truth labels. The statistics of the two datasets after preprocessing are shown in Table 4.1.

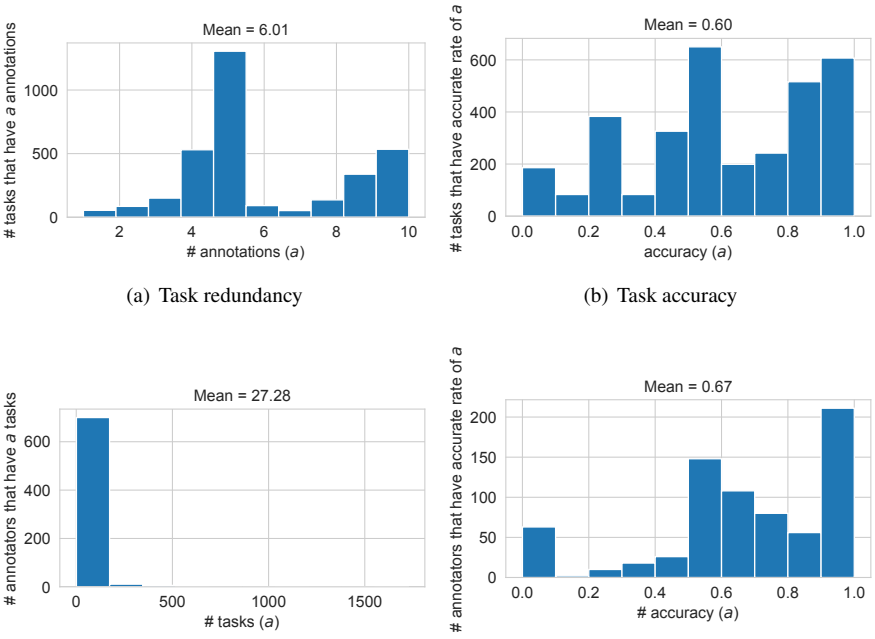
Table 4.1: Statistics of two crowdsourcing datasets.

Data set	CS2010	CS2011
# topic	100	25
# task (=# rel + # nonrel)	3,275 (1,775 + 1,500)	711 (589 + 122)
# annotator	722	181
# annotation	18,479	2,181

To understand the annotation distribution and quality of the two datasets, we are interested in: (1) how many redundant crowd annotations are collected for each task (*task redundancy*), (2) how accurate the crowd annotations are for each task (*task accuracy*), (3) how many annotations each annotator gives (*annotator redundancy*), and (4) how accurate each annotator is (*annotator accuracy*). Formally, we define the *task redundancy* of task i as the number of its crowd annotations, denoted by M_i; following [100], we define the *task accuracy* of task i as the accurate rate of its crowd annotations, denoted by $\frac{\sum_{j=1}^{M_i} \mathbf{1}(y_i^j = y_i^*)}{M_i}$; we define the *annotator redundancy* of annotator j as the number of crowd annotations he or she gives, denoted by N_j; following [100], we define the *annotator accuracy* of annotator j as the accurate rate of his or her crowd annotations, denoted by $\frac{\sum_{i=1}^{N_j} \mathbf{1}(y_i^j = y_i^*)}{N_j}$. We plot histograms for task redundancy, task accuracy, and annotator redundancy in Figure 4.2 and 4.3. Overall, the quality of the crowd annotations of CS2011 is better than CS2010.

Synthetic data

In order to fully understand the ability of our model in terms of modelling task true labels, task difficulties and annotator competences, we generate synthetic data that contain these corresponding ground truth values. Besides, we can vary the noise level and redundancy level of annotation to study their impact on the performance of our



(a) Task redundancy

(b) Task accuracy

(c) Annotator redundancy

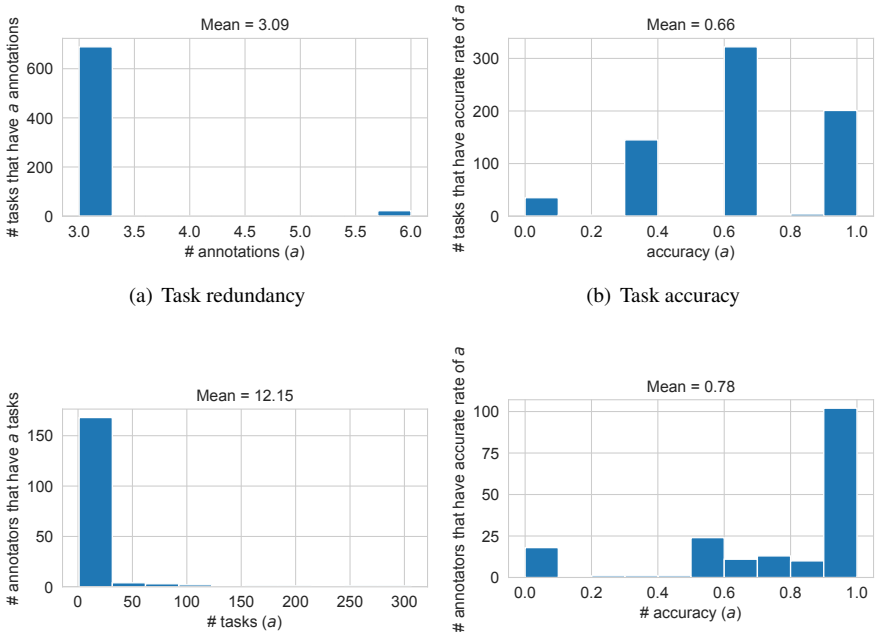
(d) Annotator accuracy

Figure 4.2: Annotation distribution of CS2010.

model. We simulate the annotation process based on the following assumptions. First, we assume that tasks are generated from two 2-dimensional Gaussian distributions: $\mathcal{N}(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}_{+1})$ for the positive class and $\mathcal{N}(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}_{-1})$ for the negative class, where $\boldsymbol{\mu}_{+1} = [1, 0]^{\mathrm{T}}$, $\boldsymbol{\Sigma}_{+1} = \boldsymbol{I}$, $\boldsymbol{\mu}_{-1} = [0, 1]^{\mathrm{T}}$, and $\boldsymbol{\Sigma}_{-1} = \boldsymbol{I}$. Second, we assume that there is a 1-dimensional Gaussian noise $\mathcal{N}(0, \sigma_i^2)$ for a task i, and the variance σ_i^2 is determined by the distance between the task vector and the mean vector of its class. Third, we assume a power law distribution $\mathcal{P} = \{\frac{1}{Z}\frac{1}{j^{\alpha}} \mid j = 1, \ldots, M\}$ where Z is a normalization factor, from which to select an annotator to annotate a task; specifically, we set $M = 50$, $\alpha = 1.5$ to make sure around 10% annotators annotate 80% tasks because it is empirically shown more than 80% of the tasks are annotated by only 10% of the workforce [78]. We also assume a Gaussian noise $\mathcal{N}(0, \sigma_j^2)$ for an annotator, and the variance σ_j^2 is determined by a controlled hyperparameter interval $\mathcal{U}$. Last, we assume the number of annotations for each task is controlled by another hyperparameter β. The data generation procedure is in Algorithm 10.

4.5.3 Baselines

Our model use a Gaussian Process for task correlation and multiple Gaussian distributions for the difficulty of tasks and the competence of annotators. It allows us to integrate prior knowledge on tasks and model the label bias of tasks, the difficulty of tasks, the label bias of annotators and the competence of annotators. In order to fully investigate the properties of our model, we compare our model with the following baselines:



(a) Task redundancy

(b) Task accuracy

(c) Annotator redundancy

(d) Annotator accuracy

Figure 4.3: Annotation distribution of CS2011.

Majority voting (MV) Majority Voting aggregates crowdsourcing annotations in a straightforward way. It infers the true label by the majority vote of annotations without considering task difficulty or annotator competence.

Multi-annotator competence estimation (MACE) [74] MACE models the latent true label of a task, the annotator competence and the task difficulty. It assumes a different annotation generation process from our MAGP model. We take MACE as a representative model of all the probabilistic graphical models for crowdsourcing as its assumption on parameters is simple enough and is applicable to most scenarios. Moreover, it is solidly implemented, well documented, and publicly available.

Gaussian process majority voting (GPMV) We propose this model as an extension of MV. GPMV assumes a GP prior over the latent true labels. It can be viewed as a vanilla Gaussian process classification model. Same to MV, it only models the latent true labels, not task difficulty or annotator competence.

Likelihood (LK) We also propose a light-weight version of the MAGP model without considering the Gaussian process prior. We name it LK as we only consider the likelihood part of the MAGP model, i.e., assume the latent variables $\{f_1, \ldots, f_N\}$ do not conform to a Gaussian process prior, and consequently Equation (4.9) does not hold any more. The LK model assumes $\{f_1, \ldots, f_N\}$ are independent variables that indicate true labels of tasks, and a crowd annotation is generated by adding a Gaussian noise of the task and a Gaussian noise of an annotator. LK is formally expressed as Equation (4.15), where model parameters of $\boldsymbol{f}$ and $\boldsymbol{\theta}$ are learned by maximizing the log likelihood of the observed data using gradient descent. LK is able to model the latent true

Algorithm 10: Synthetic data generation procedure.

Input: Annotation noise level $\mathcal{U} \in \{[0, 2), [2, 4), [4, 8), [8, 12)\}$; Annotation redundancy level $\beta \in \{3, 5, 7\}$

Output: Input features, true labels, and pseudo crowdsourcing annotations, denoted by $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{Y})$.

for $i \in [1, N]$ **do**
  Sample a class from $\{-1, 1\}$, denoted by y_i;
  Sample a point denoted by $\boldsymbol{x}_i$ from the Gaussian distribution of the corresponding class ($\mathcal{N}(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}_{+1})$ or $\mathcal{N}(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}_{-1})$);
  Denote the variance of the noise of task i by $\sigma_i^2 = (\boldsymbol{x}_i - \boldsymbol{\mu}_i)^2$;
for $j \in [1, M]$ **do**
  Uniformly sample a value denote by σ_j^2 from $\mathcal{U}$ as the variance of the noise of annotator j;
Use a normalized power law distribution $\mathcal{P} = \{\frac{1}{Z}\frac{1}{j^\alpha} \mid j = 1, \ldots, M\}$ and denote the probabilities that annotators are selected to annotate a task;
for each task $i \in [1, N]$ **do**
  Randomly sample β annotators from distribution $\mathcal{P}$;
  for each annotator j in the β annotators **do**
    Generate an annotation y_i^j through $\mathcal{N}(y_i, \sigma_i^2 + \sigma_j^2)$;

label of each task, the difficulty of each task, and the competence of each annotator.

These baselines model the aforementioned factors in crowdsourcing annotations in different ways. By comparing our model with these baselines, we can better understand the pros and cons of our model. Table 4.2 makes a comparison of these methods. Label prior denotes what prior distribution the method uses or whether it uses no prior at all, true label, annotator competence and task difficulty denote whether the method models these aspects of crowdsourcing.

Table 4.2: Comparison of our method and the baselines in terms of label prior, true label, task difficulty and annotator competence.

Method	Label prior	True label	Task difficulty	Annotator competence
MV	✗	✓	✗	✗
MACE	✗	✓	✓	✓
LK	✗	✓	✓	✓
GPMV	GP	✓	✗	✗
MAGP	GP	✓	✓	✓

4.6 Results and Analysis

4.6.1 Label Inference for Observed Tasks

This first experiment answers RQ1. We study whether the proposed model and its variations, GPMV and LK, can correctly infer the latent true labels. Furthermore, we compare the proposed model to MV and one of the state-of-the-art probabilistic graphical models MACE [74]. we are interested in whether modelling task difficulty and annotator competence can help to infer the latent true labels.

Label Quality

For label quality, we compare methods that model the true labels of tasks including our proposed model MAGP, and its simplifications GPMV and LK, as well as state-of-the-art approaches MACE and MV. LK, MACE and MV only needs crowd annotations as input. While MAGP and GPMV also incorporates axillary information of tasks to represent tasks as vectors, and we use lexical, semantic, and ranking features for task representation as this setting shows the best performance in Section 4.6.4.

We employ *F1* to evaluate the performance of the different methods. F1 takes the imbalance of two classes of the dataset into consideration and calculates the harmonic mean of precision and recall, where precision measures how many percent of label that are inferred as relevant are actually relevant, and recall measure how many percent of relevant labels are correctly inferred.

Table 4.3 shows the accuracy and F1 of the aforementioned methods in terms of inferring true labels. First, when comparing the MAGP with the state-of-the-art models, MV and MACE, it is found that MAGP and GPMV have higher F1 scores on CS2010, while MACE gets a higher F1 on CS2011. Both MAGP and MACE perform better than MV. Then we compare MAGP with its simplification, GPMV and LK, where LK lacks the GP prior and MVGP lacks modelling of multiple annotations per task. It is found that compared to MAGP, LK decreases more than GPMV. One possible reason is that even though multiple crowd annotations are reduced one annotation by MV it still provides enough information to infer the true labels; while ignoring axillary information of tasks does lose useful information of task correlation.

To sum up, the proposed MAGP model performs comparable with one of the state-of-the-art models MACE, and both are better than MV; and it is the modelling of task axillary information that helps to improve the performance of MAGP.

Table 4.3: Inferring the true labels for observed tasks (F1).

$F1$	CS 2010	CS 2011
MV	0.808	0.702
MACE	0.808	0.732
LK	0.808	0.702
GPMV	0.849	0.702
MAGP	0.848	0.707

4.6.2 Label Prediction for New Tasks

The second experiment answers RQ2. In this experiment we train MAGP and GPMV on a subset of the crowd annotations, then we predict relevance labels on the rest of the tasks. The ratio of task numbers in the two datasets is $1:4$. We use F1 to evaluate model performance.

Table 4.4 shows the the performance of MAGP and GPMV. Overall, the two models have similar performance and MAGP is slightly better than GPMV. Unlike generic probabilistic graphical models like MACE, which can only infer true labels for existing crowd annotations, MAGP and GPMV can also predict labels for new tasks. This advantage can help to select tasks during task distribution on crowdsourcing platforms.

Table 4.4: Predicting labels for new tasks.

$F1$	CS 2010	CS 2011
GPMV	0.652	0.848
MAGP	0.658	0.848

4.6.3 The Effect of Annotation Quality and Annotation Redundancy

This experiment answers RQ3. We study the effect of annotation quality and annotation redundancy on model performance. We use the synthetic data introduced in Section 4.5.2. We test four levels of annotation quality in $\{[0,2),[2,4),[4,8),[8,12)\}$, and three levels of annotation redundancy in $\{3,5,7\}$. Each of the settings is repeated 10 times and the averaged accuracy is reported.

Figure 4.4 shows the heat map of accuracy of the MAGP model under different values of annotation quality and annotation redundancy. We can see that the annotation quality has a larger effect than the annotation redundancy on the performance of the MAGP model. When the annotation noise is 0, which means the noise varies from 0 to 2 for all the annotators, the accuracy reaches 0.91–0.92; when the annotation noise is from 4 to 8, the accuracy decreases sharply to 0.85–0.88. The accuracy does not change much when increasing the annotation redundancy. This finding indicates that in practical crowdsourcing experiments one should focus on attracting annotators of high quality instead of spending budget on multiple annotations for each task.

Figure 4.5 further examines the performance of different models with varying annotation quality and annotation redundancy, respectively. MAGP and GPMV perform better than MV, MACE, and LK. The difference between MAGP and GPMV with the other models is that they both assume a Gaussian process prior on the latent variables of relevance labels, which indicates that auxiliary information of tasks helps to improve accuracy. Besides, the performance of MAGP and GPMV do not decrease much with annotation noise increasing.

To sum up, the annotation quality has a larger effect on model performance than the annotation redundancy.

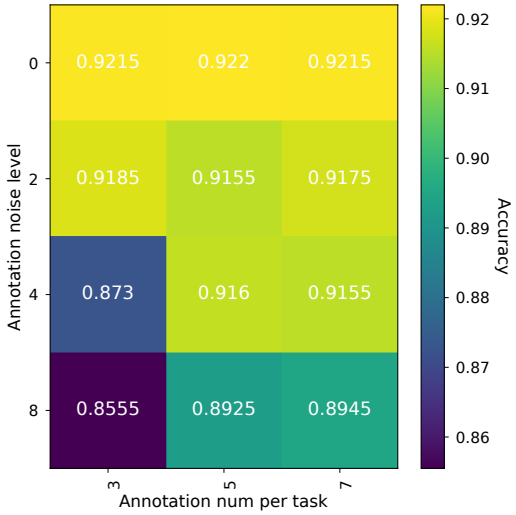


Figure 4.4: Accuracy of MAGP under different annotation quality and annotation redundancy.

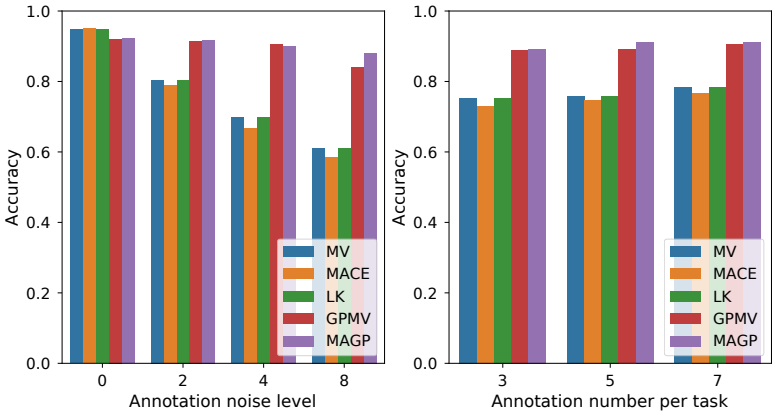


Figure 4.5: Accuracy of MAGP and baselines under different annotation quality and annotation redundancy.

4.6.4 The Effect of Different Features in Task Representation

This experiment answers RQ4. We investigate the effect of different types of features in task representation. As the MAGP model can incorporate multiple types of auxiliary information of tasks through covariance function, we propose to use features from various resources, they are lexical features, semantic features and ranking features. Then we apply the MAGP model on all of the possible different combinations of features. We use F1 to evaluate model performance.

The results are shown in Table 4.5. To sum up, the semantic features and their combinations with other features perform the best, indicating that the use of the BERT

model helps to represent queries and documents for better relevance prediction.

Table 4.5: The effect of prior knowledge.

Features	$F1$
Lexical	0.691
Ranking	0.683
Semantic	0.706
Lexical+Ranking	0.692
Ranking+Semantic	0.681
Lexical+Semantic	0.705
Lexical+Ranking+Semantic	0.705

4.7 Conclusions and Future Work

In this chapter, we study the problem of relevance inference from noisy annotations to answer **RQ3** introduced in Chapter 1. We propose a new annotation generation process modelled by a Bayesian generative model called multi-annotator Gaussian process. This defines a novel data generation process. To be specific, our model employs a Gaussian process prior on the latent true labels of each tasks and learns the correlation among tasks; further, it employs Gaussian noise on each annotator and on each task to learn task difficulty and annotator competence.

The experimental results show that our model is comparable with baselines. By comparing our method to its simplified versions as baselines we demonstrate that both the prior part and the likelihood part contribute to the improvement of the inferred label quality. Further, both the annotation quality and redundancy both impact the performance of the proposed model, and annotation quality dominates the impact.

Further, we leave as future work the impact of different covariance functions, the learning of annotators' bias, which could potentially model spammers or adversarial annotators, as well as the bias per task.

In the next chapter, we will leverage existing test collections and study how to optimize the configuration of retrieval systems for any effectiveness measure.

Part II

Optimisation

5 Optimizing Retrieval Systems

So far, we have studied how to actively select documents for the construction of a test collection and the evaluation of retrieval systems (Chapter 2). We have also studied how to determine the stopping point of document selection for constructing test collections that balance the cost of assessing document relevance and the gain of identifying relevant documents (Chapter 3). Later, we have investigated the issue of leveraging crowdsourcing techniques to acquire relevance assessments of the selected documents and aggregating the noisy crowdsourcing assessments for the construction of high quality test collections (Chapter 4). In this chapter, we leverage existing test collections to optimize the configuration of retrieval systems using Bayesian optimization (BO) techniques. We aim to answer the following research question asked in Chapter 1:

RQ4 How can we optimize the configuration of retrieval systems with regard to effectiveness measures on the basis of existing test collections?

5.1 Introduction

The effectiveness of IR systems heavily depends on a large number of hyperparameters that need to be tuned. Hyperparameters range from the choice of different system components, e.g., stopword lists, stemming methods, retrieval models, to model parameters, such as the k_1 and b values in BM25, the number of top-ranked documents to consider in pseudo-relevance feedback, and the number of query expansion terms.

Retrieval performance is rather sensitive to hyperparameter tuning. Zhai et al. [177] demonstrated this sensitivity for the smoothing parameters in language models, and Trotman et al. [159] for parameters in BM25. Automatic techniques for optimizing model hyperparameters have attracted the attention of the research community in recent years [58, 64–66, 111, 136, 143]. However, hyperparameters are usually optimized in isolation, while their mutual dependencies are, to a great extent, unexplored [7, 90]. Ferro and Silvello [56] recently examined this mutual dependency between choices of stopword lists, stemmers and retrieval models, and concluded that hyperparameter interactions have a strong effect on system performance.

We name the space that the possible values of all hyperparameters forms the *configuration space* or the *hyperparameter space*, and the two terms are used interchangablely. *Grid search* has been the most widely used strategy for automatic joint optimization of hyperparameters in IR [161]. Grid search is easy to implement, parallelization is

trivial, and it is reliable in low dimensional spaces [17]. However, grid search suffers from the *curse of dimensionality* because the number of configurations grows exponentially with the number of hyperparameters [16].

To reduce the number of candidate configurations to be explored, researchers often apply grid search on one dimension (or hyperparameter) at a time, while fixing the values of the remaining hyperparameters until all of them have been traversed. However, this approach is not guaranteed to find a global optimum. Bergstra et al. [17] instead propose *random search* as a more efficient search algorithm. Random search generates candidate configurations drawn uniformly from the same configuration space as would be spanned by a regular grid. Bergstra et al. [17] show that random search is more efficient than grid search in high-dimensional spaces because objective functions of interest often have a low effective dimensionality, i.e., they are more sensitive to changes in some dimensions than others [26]. Nevertheless, random search remains agnostic to the effect that a δ-step in the hyperparameter space would have to the effectiveness of an IR system. Quantifying this effect could lead to more efficient search strategies.

BO has risen as a promising framework for efficient and effective search in the candidate configuration space [18, 19, 75, 144, 149, 156, 176]. Under the BO framework, one does not need to explicitly specify the objective function; what is only necessary is the ability to query this function and get an observation. In the case of IR, one does not need to analytically express system effectiveness as a function of the hyperparameters, but only observe the effectiveness of a system configuration in terms of an evaluation measure, e.g., average precision. BO uses a surrogate model to approximate system effectiveness. Taking a prior belief over this surrogate model allows BO to sequentially refine it. Further, the surrogate model, which indicates a probability distribution over the possible system effectiveness functions, allows one to design different strategies for selecting the next configuration to be tested, to exploit hyperparameter subspaces that have shown to contain high performance configurations, or to explore hyperparameter subspaces with high potential [144].

In this chapter, we use BO to automatically optimize the retrieval system hyperparameters. Information retrieval effectiveness, however, as a function of the hyperparameter space exhibits high irregularity. Furthermore, hyperparameters can be continuous, or categorical [52], with the latter contributing most of the irregularity of the objective function. To tackle this we model the effect of a δ-step in the hyperparameter space to the effectiveness of the IR system, by suggesting to use different similarity functions (covariance functions) for continuous and categorical hyperparameters, and examine their ability to effectively and efficiently guide the search in the hyperparameter space. We compare BO to manual tuning, grid search, and random search using TREC collections, and demonstrate that BO is able to find better configurations in terms of retrieval effectiveness when all methods are granted the same computational budget. To the best of our knowledge, this is the first work that uses BO to find optimal configurations of retrieval systems.

Therefore, the main contributions of this chapter are the following:

1. We propose the use of BO for retrieval system configuration;
2. We decompose the components of BO that affect the effectiveness of the method and suggest an instantiation of it that fits the IR hyperparameter space; and
3. We demonstrate the effectiveness of the method in building IR systems and explain

the reason in terms of optimization behavior.

5.2 Related Work

We first discuss prior work on hyperparameter optimization in IR. Next, we give an overview of BO methods and the domains they have been applied to.

5.2.1 Hyperparameter Optimization in Information Retrieval

Optimizing the hyperparameters of a retrieval system is inevitable in order to achieve optimal performance [161]. Different methods have been proposed in the literature for automatically optimizing individual components of a retrieval system [58, 64–66, 111, 136, 143, 157, 179].

Taylor et al. [157] use a gradient descent method to optimize the parameters of ranking functions like BM25F. Their approach, however, makes the assumption that the cost functions must be differentiable. Bigot et al. [20] propose a method for hyperparameter optimization on a per-query basis. However, this method is hard to generalize as it only works on queries already seen in the training set. Deveaud et al. [48] cast the problem of system configuration optimization as a ranking problem and use learning to rank approaches to select the best combination of hyperparameters for IR systems. The drawback is that all the configurations to be ranked must be run in advance for training and the element number of configurations (e.g., about 10,000 in this chapter) grows exponentially with the space dimension. The work of Bigot et al. [20] and Deveaud et al. [48] is orthogonal to the work in this chapter. In both aforementioned works all candidate configurations need to be considered and the focus lies in finding the best one of them for each query. Instead, in this chapter we propose a search strategy that avoids considering all candidate configurations and focuses search on the most promising sub-spaces of the hyperparameter space.

5.2.2 Bayesian Optimization

The problem of hyperparameter optimization appears in many machine learning applications, and lately it has attracted a wide interest in that community [18, 19, 75, 144, 149, 156, 176]. BO has emerged as a promising framework for efficiently identifying effective configurations. Popular BO approaches include sequential model-based optimization [149], sequential model-based algorithm configuration [75], tree-based Parzen estimator [19] and multi-task BO [156]. BO methods have been applied successfully in many tasks [155, 166, 176]. For example, BO can find better hyperparameters for deep neural networks in MNIST digit recognition and CIFAR-10 object recognition [155]. Also, by applying the tree-based Parzen estimator in feature selection, basic machine learning algorithms like logistic regression and support vector machine are proven to outperform state-of-art neural network models in topic classification and sentiment analysis tasks in natural learning processing (NLP) [176]. As an important surrogate model, Gaussian process (GP) models as well as its covariance functions have also been studied extensively [76, 155]. These successful applications have inspired us to use BO to optimize retrieval systems.

5.3 Preliminaries

5.3.1 Bayesian Optimization

The BO framework provides a mechanism to sequentially search for the global optimum x of an objective function $f(x) : \mathbb{X} \rightarrow \mathbb{R}$. There are two key components in BO [144]. The first is a probabilistic *surrogate model* used to predict the objective function value y given a point x. For every x, there is a random variable y, whose distribution $p(y \mid x)$ is given by the surrogate model. One example of the predictive distribution $p(y \mid x)$ can be observed in the top panels of Figure 5.1. In this example the surrogate model is a GP (described in Section 5.3.2). The black curve depicts the original objective function (for instance mean average precision), while the x-axis in the figure represents a 1-dimensional hyperparameter space, e.g., the parameter μ of a language model with Dirichlet smoothing. The predictive distribution of y can then be used to construct an *acquisition function*. An acquisition function is a policy for selecting the sequence of points $\{x_1, x_2, \ldots, x_i, \ldots\}$, i.e., a mechanism to select the next configuration x_{n+1} to test given $D_{1:n} \triangleq \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. As the acquisition function x is usually a closed-form expression of hyperparameters, it is easier to be optimized than the original objective function. The bottom panel of Figure 5.1 demonstrates an acquisition function. The acquisition function values for different configurations (i.e., along the x-axis) dictate the potential of a certain configuration to yield a high objective function value.

The second component is the *objective function* itself, a function of the target model requiring hyperparameter optimization. In our case the objective function can be any retrieval effectiveness measure, such as average precision, normalized discount cumulative gain and so on. Computing the objective function for different system configurations is the most time-consumptive step [19].

The entire process of BO is demonstrated in Algorithm 11. The algorithm stops when the computational budget is exhausted. Figure 5.1 uses an 1-dimensional continuous function to illustrate the optimization process, and demonstrates it at round 4 (i.e., after the first 3 configurations have been tests) at the left-most panels, and at round 10 at the right-most panels.

5.3.2 Gaussian Process

The Gaussian Process (GP) is the most commonly-used surrogate model in BO [119]. The key hypothesis underlying GP is that any finite set of $\{y_i\}$ follows a multivariate Gaussian distribution. There is an analytic expression for the joint distribution $p(y_{1:n} \mid x_{1:n}) \sim \mathcal{N}(0, K)$, with K being the covariance matrix:

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \tag{5.1}$$

and $k(x_i, x_j)$ being a covariance function. The conditional distribution of the objective function value given the sequence of past observations and a new point x_{n+1} is $p(y_{n+1} \mid x_{n+1}, D_{1:n}) \sim \mathcal{N}\left(\mu_n(x_{n+1}), \sigma_n^2(x_{n+1})\right)$, where $\mu_n(x_{n+1})$, and $\sigma_n^2(x_{n+1})$ are the mean and the variance of the posterior distribution, respectively. They are

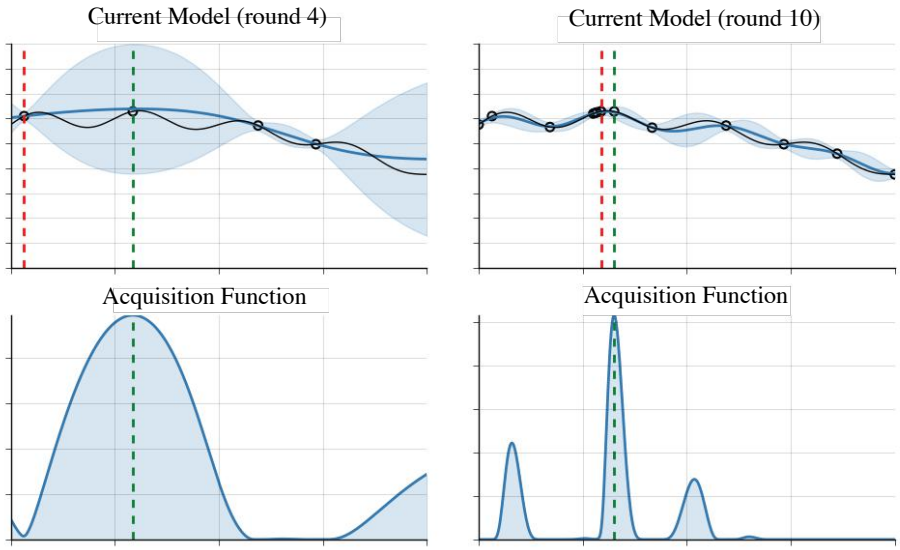


Figure 5.1: The optimization process of BO. The upper panel describes the current model, and the bottom panel describes the acquisition function. The black curve is the actual objective function, the black dots are the observed values of the objective function. The red vertical line denotes the best value among the observed points, the green vertical line denotes the next point that BO choose. The light blue line and shade are the estimated function values and variances.

Algorithm 11: Bayesian Optimization

Input: Surrogate model $\mathcal{M}$, acquisition function $\alpha_{\mathcal{M}}$, objective function f, input space X.

Initially sample k points $\{x_1, x_2, ..., x_k\}$ from X, query f to get $\{y_1, y_2, ..., y_k\}$;
Update $\mathcal{M}$;
for $n = 1, 2, ...$ **do**
    Select $x_{n+1} \leftarrow \arg\max_x \alpha_{\mathcal{M}}(x; D_n)$;
    Calculate $y_{n+1} \leftarrow f(x_{n+1})$;
    Augment $D_{n+1} = \{D_n, (x_{n+1}, y_{n+1})\}$;
    Update $\mathcal{M}$;
Select the best y from D;

defined as

$$\mu_n(x_{n+1}) = k^T K^{-1} y_{1:n} \tag{5.2}$$

$$\sigma_n^2(x_{n+1}) = k(x_{n+1}, x_{n+1}) - k^T K^{-1} k. \tag{5.3}$$

where $k = [k(x_{n+1}, x_1), k(x_{n+1}, x_2), \ldots, k(x_{n+1}, x_n)]^T$.
The covariance function models the effect of taking a δ-step in the hyperparameter space to the objective function. In Section 5.4.2 we suggest covariance functions we

believe to be appropriate for the continuous, and categorical hyperparameters of a retrieval system.

5.3.3 Acquisition Function

A good acquisition function is one that finds an optimal trade-off between exploration and exploitation in the hyperparameter space on the basis of the application at hand. In practice, an acquisition function balances between configurations for which the predicted function value $f(x)$ is high (exploitation) and configurations for which the predicted variance $\sigma(x)$ is high (exploration) [144]. In this chapter, we consider three acquisition functions: probability of improvement (PI), expected improvement (EI) [51] and upper confidence bound (UCB), defined as follows:

$$\begin{aligned}
\alpha_{PI}\left(x \mid D_n\right) &\triangleq P\left(y > y^*\right) \\
&= \Phi\left(\frac{\mu_n\left(x\right) - y^*}{\sigma_n\left(x\right)}\right) && (5.4) \\
\alpha_{EI}\left(x \mid D_n\right) &\triangleq \mathbb{E}\left(\max\left(y - y^*, 0\right)\right) \\
&= \left(\mu_n\left(x\right) - y^*\right)\Phi\left(\frac{\mu_n\left(x\right) - y^*}{\sigma_n\left(x\right)}\right) \\
&\quad + \sigma_n\left(x\right)\phi\left(\frac{\mu_n\left(x\right) - y^*}{\sigma_n\left(x\right)}\right) && (5.5) \\
\alpha_{UCB}\left(x \mid D_n\right) &\triangleq \mu_n(x) + \beta\sigma_n(x)\,, && (5.6)
\end{aligned}$$

where $\alpha(\cdot)$ is the acquisition function, $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative distribution function and the probability density function of the standard normal distribution, respectively. y^* is a target value which is often set to $\max\left(\{y_i\}_{i=1}^n\right)$ [167]. β is a hyperparameter that can be set according to some theoretically motivated guidelines.

A drawback of PI, intuitively, is that it is pure exploitative. Configurations that have a high probability of their effectiveness being infinitesimally greater than y^* will be drawn over configurations that offer larger gains but with less certainty. EI, on the other hand, considers the magnitude of the improvement a configuration can potentially yield, instead of PI. In UCB it is the parameter β that controls the trade-off between exploration and exploitation. These three acquisition functions are point-wise, that is, they only care about the improvement over y^* on each single point. There are also entropy-based acquisition functions that make use of what has been observed about the objective function to pick up the most informative point, which will be left for the future study.

5.3.4 Initialization

The first posterior distribution of the surrogate model can be obtained by using the predefined prior distribution to draw the first hyperparameter point, i.e., the first configuration. This strategy runs the risk of getting stuck in a local optimum, since it heavily depends on the first point that will be used. A more general strategy, which we follow in this chapter is to sample a set of configurations randomly in the search space so that the surrogate model can observe the overall "landscape" of configurations. In this chapter we test three sampling methods: Latin hypercube sampling (Latin), uni-

form random sampling (Uniform), and Sobol sequence based sampling (Sobol) [69].

5.3.5 Selecting the Optimal Configuration

At the end of the optimization process one needs to decide which configuration is the optimal one. The natural choice is to select the configuration with the best observed effectiveness. However, this strategy runs the risk of overfitting the observed objective function values. A different strategy is to select the configuration with the highest predicted value, instead of the actual observed value, which can accommodate noisy outputs. The two strategies are called *incumbent* and *latent*, respectively [69], and we test both in this chapter.

5.4 Methodology

In this section we first elaborate the hyperparameter optimization process, then we focus on the particular characteristics of the retrieval systems hyperparameter space and discuss the covariance functions we consider appropriate for this space.

5.4.1 Hyperparameter Optimization Process

Following the BO framework, we have two major modules in our algorithmic pipeline, the IR module and the BO module (see Figure 5.2). The IR module tackles the conditional hyperparameters, and computes the objective function value y_n, given a hyperparameter configuration x_n. The BO module adds (x_n, y_n) into the sample set, updates the posterior distribution of the surrogate models, selects the next hyperparameter configuration x_{n+1}, and passes it back to IR module.

5.4.2 Hyperparameter Structure and Covariance Functions

Snoek et al. [149] suggest that it is the choice of the covariance functions that has the strongest effect on the performance of BO. In this chapter we only consider a simple case, the *stationary covariance function*, which is defined as a function of $|\boldsymbol{x} - \boldsymbol{x'}|(= r)$. The *mean square differentiability* of a stationary covariance function around 0 determines the smoothness properties of the samples drawn from the corresponding GP [129], which essentially dictates the expected response in the objective function if a δ-step is taken in the hyperparameter space.

The *squared exponential covariance function* (SE) is a widely made choice for smooth objective functions; it is mean square differentiable at any order and thus very smooth. It is defined as follows:

$$K_{SE}(\boldsymbol{x}, \boldsymbol{x'}) = \exp\left(-\frac{r^2}{2}\right), \tag{5.7}$$

where $\boldsymbol{x}$ and $\boldsymbol{x'}$ are two points in the hyperparameter space, and r is the distance between $\boldsymbol{x}$ and $\boldsymbol{x'}$.

The *Matérn1 covariance function* (Matérn1) is appropriate to model rough objective functions, as it is only first-order mean square differentiable. Martén1 is defined as follows:

$$K_{Matérn1}(\boldsymbol{x}, \boldsymbol{x'}) = \exp(-r)\,. \tag{5.8}$$

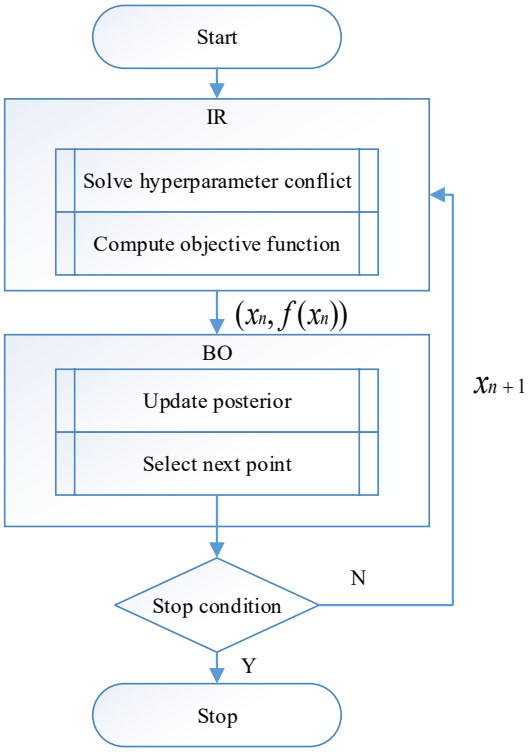


Figure 5.2: Hyperparameter optimization architecture.

Selecting a covariance function requires examining the hyperparameter space of an IR system. Retrieval model parameters, e.g., the smoothness parameter λ in the Jelinek-Mercer smoothing method, or the parameter b in the Okapi BM25 model, are typically continuous. On the other hand, the choice of the retrieval model itself can be expressed with a categorical hyperparameter; the use or not of relevance feedback algorithms is also categorical, as is the choice of a stopword list, or a stemmer. The default distance used in either SE covariance function or Matérn1 covariance function is Euclidean distance, defined as follows:

$$r_E \triangleq \sqrt{\sum_{d=1}^{N} (x_d - x_d{}')^2}\,. \tag{5.9}$$

r_E would treat categorical hyperparameters as ordinal, whereas this should not be the case. Therefore, considering the heterogeneity of the hyperparameters of IR systems, Euclidean distance is not a good fit. Instead, inspired by [75], we use *Hamming distance* for categorical dimensions, while we use *Euclidean distance* for continuous or discrete dimensions. The *Hamming distance* is defined as follows:

$$r_H \triangleq \sum_{d=1}^{N} \left(1 - \delta\left(x_d, x_d{}'\right)\right)\,, \tag{5.10}$$

where δ is the Kronecker delta function (equalling one if its two arguments are identical and zero otherwise).

In order to accommodate both continuous and categorical hyperparameters, we combine the Euclidean distance and the Hamming distance into a mixture distance, the *Hamming Euclidean mixture distance* (HE distance). HE treats continuous and categorical hyperparameters differently; it is defined as follows:

$$r_{HE} \triangleq \sqrt{\sum_{d \in \text{con}} \left(x_d - x_{d'}\right)^2 + \sum_{d \in \text{cat}} \left(1 - \delta\left(x_d, x_{d'}\right)\right)}, \tag{5.11}$$

where *con* is the set of continuous dimensions and *cat* the set of categorical dimensions. It is easy to prove that HE is a distance (or metric). By replacing the original distance function r_{SE} with r_{HE}, we get two new covariance functions,

$$K_{HSE}\left(\boldsymbol{x}, \boldsymbol{x'}\right) = \exp\left(-\frac{r_{HE}^2}{2}\right) \tag{5.12}$$

$$K_{HMatérn1}\left(\boldsymbol{x}, \boldsymbol{x'}\right) = \exp\left(-r_{HE}\right). \tag{5.13}$$

Compared with SE and Matérn1, HSE and HMatérn1 are designed to handle heterogeneous spaces where the continuity property of different dimensions are not the same. To sum up, in this chapter, we test the performance of SE, HSE, Matérn1 and HMatérn1 as four representative covariance functions for continuous and categorical hyperparameters.

5.5 Experimental Setup

5.5.1 Research Questions

In the remainder of the chapter we aim to answer the following three research questions:

RQ1 What is the most critical component of the BO framework for identifying the best retrieval system configuration?

RQ2 How effective is BO in searching the configuration space, and in finding configurations that generalize across queries and collections?

RQ3 How to explain the optimization behavior of BO in terms of exploration and exploitation?

5.5.2 Implementation

We use *Pyndri* [160], a Python Interface to the *Indri* search engine [152], as the IR module in our pipeline, which is mainly decomposed to indexing, retrieval, and pseudo-relevance feedback. All three modules are considered in our optimization experiments.

The objective function can be any retrieval effectiveness measure. In this chapter we optimize for the mean average precision (MAP), normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MRR) and use *trec_eval*[1] for the computations.

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

We use *Pybo* [69] in our experiments, a Python package for BO. *Pybo* supports the default version of the SE covariance function and the Martén1 covariance function, which use Euclidean distance to measure the distance of two points; we implemented the HSE covariance function and the HMartén1 covariance function within *Pybo* ourselves.

5.5.3 Candidate Configurations

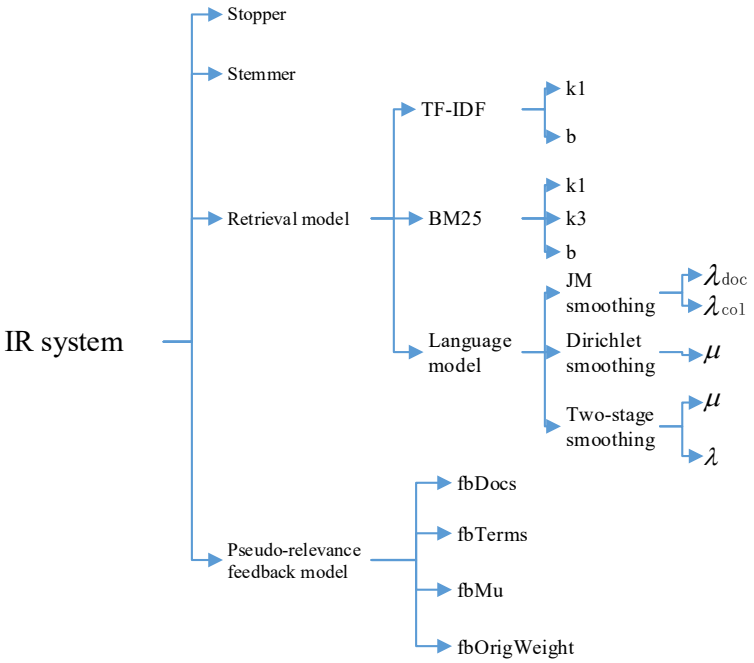
There are two major choices to make when indexing documents: the stopword list and the stemmer. Ferro et al. [56] have shown that the choice of different stopword lists, like that of Indri, Lucene, Smart and Terrier, has limited effect to the performance of a IR system; however, having a stopword list or not makes a big difference. On the other hand, indexing document collections takes a long time. Therefore for efficiency and convenience we only set two values for the categorical hyperparameter *stopper*: *true* means using the Indri stopword list and *false* means not using any stopword list. The situation is similar for *stemmer*: *true* means using Krovets stemmer and *false* means not using any stemmer.

There are three retrieval models implemented in *Indri*: the TF-IDF model with the BM25 term weighting, the Okapi BM25 model, and the Language Model. The Language Model supports three different smoothing methods, Jelinek-Mercer, Dirichlet, and two-stage smoothing. Let *rm* denote a categorical hyperparameter that represents the retrieval model type and takes values in {*tf-idf*, *bm25*, *lm-jm*, *lm-dir*, *lm-ts*}, that is the TF-IDF model with BM25 term weighting, the Okapi BM25 model, the Language Model with JM smoothing, the Language Model with Dirichlet smoothing, and the Language Model with two-stage smoothing respectively. There is a number of hyperparameters per model: k_1 and b for the TF-IDF model with BM25 term weighting, k_1, k_3, and b for the Okapi BM25 model, λ_{col} and λ_{doc} for the JM smoothing, μ for the Dirichlet smoothing, and λ and μ for the two stage smoothing. The parameters of these models lay in a continuous space.

Indri also supports pseudo-relevance feedback models. We consider a binary hyperparameter *prf* that takes the value *true* if pseudo-relevance feedback is used, and *false* otherwise. There are four hyperparameters that need to be set for the pseudo-relevance feedback models, the number of feedback documents to be considered, *fbDocs*, the number of feedback terms, *fbTerms*, the Dirichlet smoothing parameter used for the feedback document language model, *fbMu*, and the weight of the original query, *fbOrigWeight*, in the mixture model between the original query and the feedback documents. In total, we have a conditional hyperparameter space of 18 dimensions (see Figure 5.3).

5.5.4 Test Collections

We conduct our experiments on the ad-hoc test collections of TREC 5–8 and TREC Robust 2004, as well as the web test collections of TREC 2010–2012 in order to thoroughly evaluate the proposed method in diversified datasets. The details can be found in Table 5.1. CR is short for the Congressional Record documents. As we can see, ad-hoc tracks and web tracks are quite different test collections. Web tracks have much more documents and the average length of documents is also longer, indicating they are more challenging than ad-hoc tracks.



(a) Conditional hyperparameters.

Hyperparamter	Type	Values
$stopper$	Boolean	{*true*, *false*}
$stemmer$	Boolean	{*true*, *false*}
rm	Integer	{*tf-idf*, *bm25*, *lm-jm*, *lm-dir*, *lm-ts*}
k_1 (TF-IDF)	Real value	[1,2]
b (TF-IDF)	Real value	[0,1]
k_1 (BM25)	Real value	[1,10]
k_3 (BM25)	Real value	[1,10]
b (BM25)	Real value	[0,1]
λ_{doc}	Real value	[0,1]
λ_{col}	Real value	[0,1]
μ_{dir}	Real value	[0,3000]
s μ_{ts}	Real value	[0,3000]
λ_{ts}	Real value	[0,1]
prf	Boolean	{*true*, *false*}
$fbDocs$	Integer	[1,50]
$fbTerms$	Integer	[1,50]
$fbMu$	Real value	[0,3000]
$fbOrigWeight$	Real value	[0,1]

(b) Search range in *Indri*.

Figure 5.3: Conditional hyperparameters and their search ranges in *Indri*.

Table 5.1: Test collections.

TREC	Document collection	Doc number	Doc length		Topics
			Median	Mean	
TREC 5	Volume 2 & 4	524,929	340	546	251–300
TREC 6	Volume 4 & 5	556,077	326	526	301–350
TREC 7	Volume 4 & 5 \ CR	528,155	328	480	351–400
TREC 8	Volume 4 & 5 \ CR	528,155	328	480	401–450
TREC Robust 2004	Volume 4 & 5 \ CR	528,155	328	480	301–450, 601–700
TREC Web 2010	ClueWeb09	21,258,800	629	1096	51–100
TREC Web 2011	ClueWeb09	21,258,800	629	1096	101–150
TREC Web 2012	ClueWeb09	21,258,800	629	1096	151–200

5.5.5 Baselines

Manual search The first baseline is obtained by running *Indri* with its default hyperparameters. Then we manually select the best performing model among the TF-IDF model with the BM25 term weighting, the Okapi BM25 model, and the Language Model, with or without pseudo-relevance feedback.

Grid search Grid search is the most widely used method for hyperparameter tuning. For each of the five retrieval models (*tf-idf*, *bm25*, *lm-jm*, *lm-dir*, *lm-ts*) in Section 5.5.3, we generate the grid points by evenly partitioning the search space of each parameter into 20 parts and making a complete combination of all the parameters of that retrieval model. Furthermore, based on the five sub-baselines, we easily get five more sub-baselines by setting *pfb* = *true* and fixing *fbDocs* = 10, *fbTerms* = 10, *fbMu* = 2500, and *fbOrigWeight* = 0.5. For instance, the baseline (*rm=lm-dir*, *prb=true*) contains 20 search points obtained by setting μ to the values partitioning $[0, 3000]$ into 20 parts. For all the sub-baselines, *stopper* and *stemmer* are set to *true*. This is common practice in the use of grid search, and allows the reduction of the 6.6×10^{19} possible points of our hyperparameter space down to 1.8×10^4.

Random search [17] For random search, we use 3 sampling designs, Uniform, Latin, and Sobol, as introduced in Section 5.3.4. We allow its number of iteration same with BO.

5.6 Results and Analysis

5.6.1 Decompose Component Effect of Bayesian Optimization

This experiment is designed to answer RQ1. In order to study which component of BO has the strongest effect on the performance of BO in finding optimal system configurations we run a full factorial experiment, using 54 joint strategies of BO: the 3 initialization strategies $\times$ the 3 covariance functions $\times$ the 3 acquisition functions $\times$ the 2 selection strategies. The experiment was run on the Robust 2004 dataset. We measure performance on the basis of MAP and we run an analysis of variance (ANOVA) considering only the main effects of the BO components (that is we ignore any interactions between them, since we do not have enough data points for a detailed analysis).

The results of the ANOVA can be seen in Table 5.2. The important observation lays in the last column of the table; this is the p-value that designates whether a factor has a significant effect when $p(>\text{F})<0.05$, or there is not enough evidence to reach that con-

clusions otherwise. As we can observe, and in accordance with previous work [149] ANOVA suggests that the choice of the covariance function is the most important decision one needs to make when instantiating BO, at least among the components we considered in this chapter. Therefore, we vary covariance functions and fix the remaining strategies by selecting the respective best ones, that is Sobol + EI + incumbent in the following sections.

Table 5.2: The effects of different components of BO on objective function via ANOVA.

	SS	DF	F	$p(>F)$
Initialization strategy	0.000566	2.0	0.903685	0.412144
Covariance function	0.002976	2.0	4.748861	**0.013338**
Acquisition function	0.000647	2.0	1.032076	0.364368
Selection strategy	0.000543	1.0	1.734376	0.194376

5.6.2 Optimizing Retrieval Systems using Bayesian Optimization

This experiment is designed to answer RQ2. We first run the four methods on Robust 2004 or Web 2012 as the training set, then we tested the corresponding optimal configurations on TREC 5–8 or Web 2011–2012. In order to have a thorough study of BO, we further experimented on the 2-dimensional space, which is is λ and μ for the two stage smoothing, and the 18-dimensional space, which includes the complete hyperparameters introduced in Section 5.5.3, respectively.

Table 5.3 shows the performance of the best configurations found by each method. As expected, there is no big difference among the four search methods in 2-dimensional space. It is because both grid search and random search are able to cover the space in low dimensional space. However, grid search needs more budget compared with BO and random search. Therefore, BO, grid search and random search can achieve comparable performance, but BO and random search are more efficient in 2-dimensional spaces than grid search. The situation changes with the number of dimension increasing. We can see BO performs slightly better than random search, grid search and manual method in 18-dimensional space. Random search is comparable with BO in the low dimensional space (2-dimensional), but it fails in the high dimensional space (18-dimensional). However, there seems no significant difference among the four methods according to the results in Table 5.3 .

5.6.3 Optimization Behaviour of Bayesian Optimization

This experiment is designed to answer RQ3. We study how BO searches points in the search space and how different covariance functions affect this behaviour. For visualisation convenience we take a 1-dimensional and 2-dimensional hyperparameter space as examples.

The hyperparameter considered in the 1-dimensional space is μ in a language model with Dirichlet smoothing. We initialised the posterior of the surrogate model with 2 points, and iteratively searched 32 points. We also recorded the predictive function predicted by the surrogate model in round 1 and 32. The result is shown in Figure

Table 5.3: The best performing configurations for all search methods.

Method	Budget	Train A			Test A			Train B			Test B		
		MAP	NDCG	MRR	MAP	NDCG	MRR	MAP	NDCG	MRR	MAP	NDCG	MRR
2-D													
Manual	-	2415	5144	6915	1986	4446	6091	1195	2508	4084	1002	2393	4305
Grid	400	2531	5239	7028	2058	4535	6255	1216	2523	4350	1002	2395	4458
Random	4+32	2529	5239	7020	2058	4534	6341	1215	2522	4254	1000	2395	4458
BO SE	4+32	2520	5239	7022	2010	4284	6258	1212	2518	4352	998	2388	4408
BO Matérn1	4+32	2527	5239	7020	2056	4514	6314	1204	2516	4353	1003	2394	4389
18-D													
Manual	-	2755	5507	6994	2221	4817	5744	1209	2509	4242	939	2282	4019
Grid	2480	2889	5616	7248	2256	4824	6184	1404	2620	4773	922	2311	4100
Random	36+64	2692	5454	7041	2084	4776	6309	1404	2620	4773	922	2311	4100
BO SE	36+64	2908	5499	7162	2301	4705	6077	1395	2576	4860	926	2281	3933
BO HSE	36+64	2924	5549	7032	2064	4737	6273	1395	2551	5181	935	2334	3931
BO Matérn1	36+64	2925	5571	7192	2283	4781	6168	1398	2593	5268	929	2372	4140
BO HMatérn1	36+64	2747	5661	7050	2161	4864	6301	1308	2618	5502	974	2315	4160

Note: Magnitude of numbers is 10^{-4}. Train A denotes TREC Robust 2004, Test A denotes TREC 5-8, Train B denotes TREC Web 2012, Test B denotes TREC Web 2010-2011.

5.4 and 5.5, which shows the optimization behaviour of BO in 1-dimensional space $X = [0, 3000]$. We can see that the intensive red lines tend to appear near the global optimum for most datasets and measures, indicating that BO spends more efforts and exploits more near the global optimum. It is more obvious when the objective function is highly irregular like MRR. By "irregular" we mean there are many local optimums. On the right panel of Figure 5.4, we find that BO prefers exploitation near these local optimums and allows exploration in other areas, and finally it spends most efforts near the global optimum. Figure 5.5 shows the zoomed-in result on TREC 6 + MRR. The posterior of the surrogate model in round 1 does not know much about the objective function and predicts the same value for most points except the area near the two initial points. However in round 32, the predictive function can model the rough trend quite well, where several local maximums and minimums are quite consistent with the real objective function. Overall speaking, BO performance in 1-dimensional space is as good as we expect in terms of exploration and exploitation. Its preference of exploitation near the global optimum makes it a reliable optimization approach.

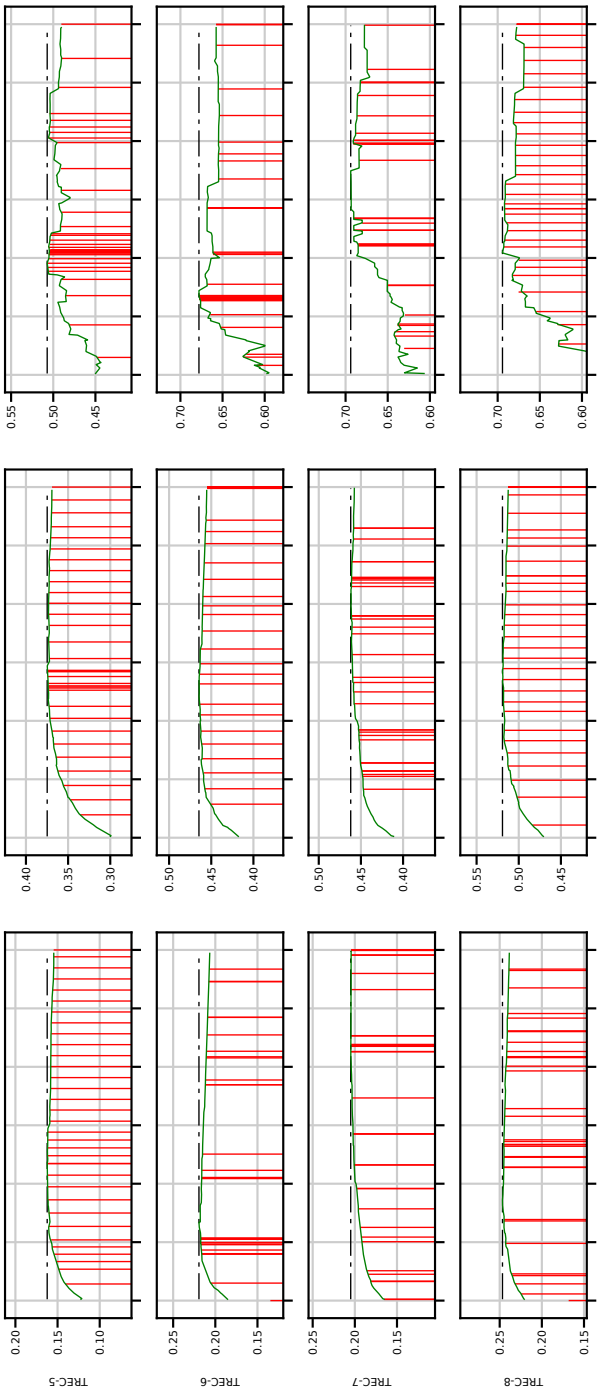


Figure 5.4: Overall optimization behaviour in a glance. Covariance function: Matérn1, objective functions: MAP, NDCG, MRR. The red vertical line associated with a number denotes BO searches which point in which round. The green curve denotes the real objective function generated by 128 devisions of the interval $[0, 3000]$. The black dot line denotes the maximum objective function value.

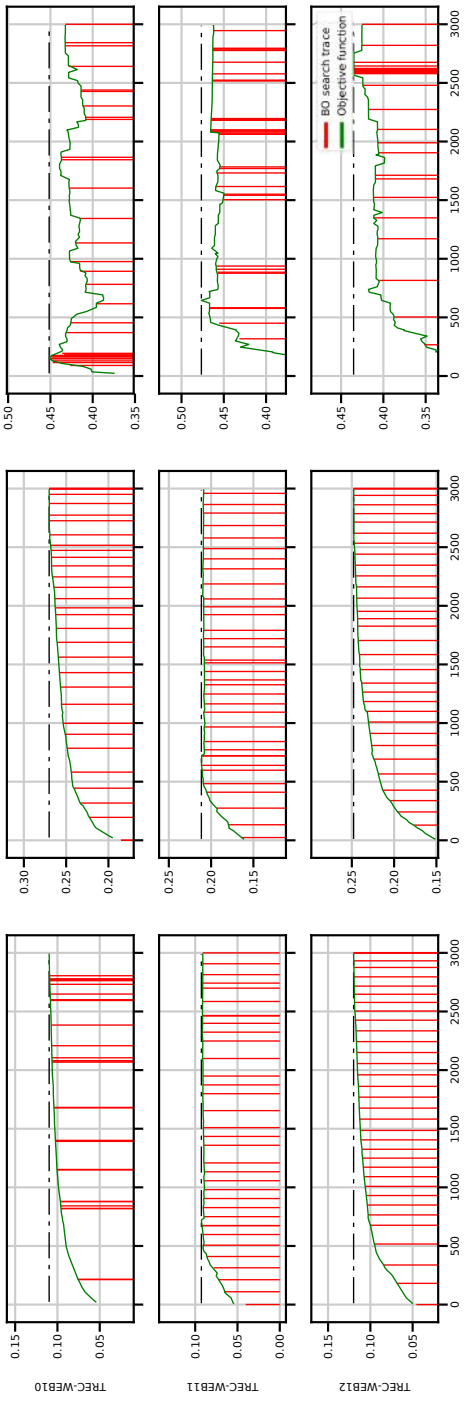


Figure 5.4: (Con't) Overall optimization behaviour in a glance. Covariance function: Matém1, objective functions: MAP, NDCG, MRR. The red vertical line associated with a number denotes BO searches which point in which round. The green curve denotes the real objective function generated by 128 devisions of the interval $[0, 3000]$. The black dot line denotes the maximum objective function value.

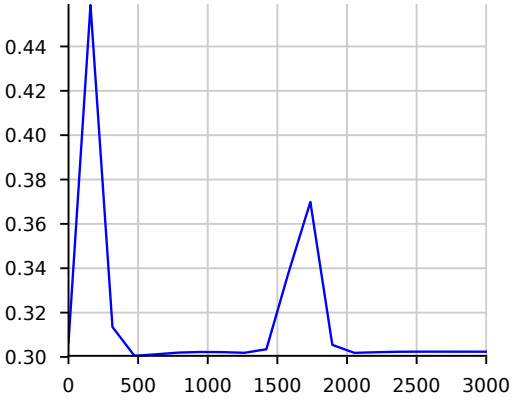
The hyperparameters considered in the 2-dimensional space are μ and λ in a language model with two-stage smoothing. We initialised the posterior of the surrogate model with 4 points, and iteratively searched 32 points. As in the 1-dimensional case, we also recorded the predictive function in round 1 and 32. In Figure 5.6 and 5.7 we plot the results of the SE covariance function and the Matérn1 covariance function respectively. The figure illustrates the optimization behavior of BO in 2-dimensional space $X = \{(\mu, \lambda) \mid \mu \in [0, 3000], \lambda \in [0, 1]\}$. The objective function is MAP, the data set is TREC-WEB12. The contour lines depict the predictive functions in round 1 and 32 on the first two panels, and the real objective function on the last panel. The red numbers denote BO searches, which point and in which round; the black dots denote the point trace of random search; and the star denotes the point of manual search.

The first two panels of Figure 5.6 shows the effect of Matérn1 on the predictive function. In round 1 the surrogate model predicts that the lower left corner has the potential of getting high values. In round 32 the predictive function is quite rugged because multiple local optimums are observed. It predicts the area around $(2800, 0.7)$ having a global optimum, which is quite consistent with the real objective function. The right panel compares the point traces of BO, random search and manual search. In the beginning, BO prefers exploration and tries to cover a large area of the search space. Later it prefers exploitation as we can see it spends more efforts near the global optimum. It is also interesting to compare the way how the two covariance functions model the real objective function. We know that the smoothness of the samples generated by the two covariance functions conforms to the order: SE >Matérn1. This is consistent with Figure 5.6 and 5.7. The real objective function in this case is quite smooth and seems to have one optimum, therefore, SE is enough to model its irregularity. As evidence we can see that more points are searched within the green contour line on the most right side in Figure 5.7 compared with Figure 5.6.

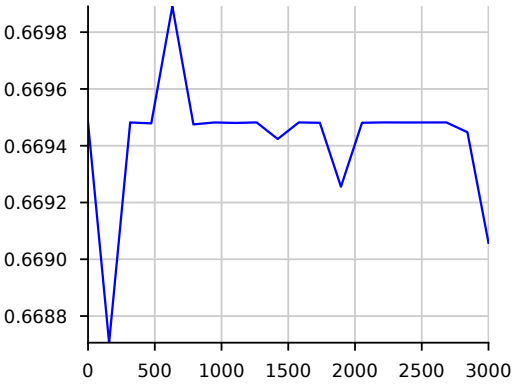
To sum up, BO balances exploitation and exploitation by spending more search budget near the global optimum or local optimums, which makes it a reliable optimization approach. This optimization behavior is affected by the covariance function. If the objective function is very irregular like MRR, Matérn1 is recommended; while SE is recommended if relative objective functions like MAP and NDCG.

5.7 Conclusion

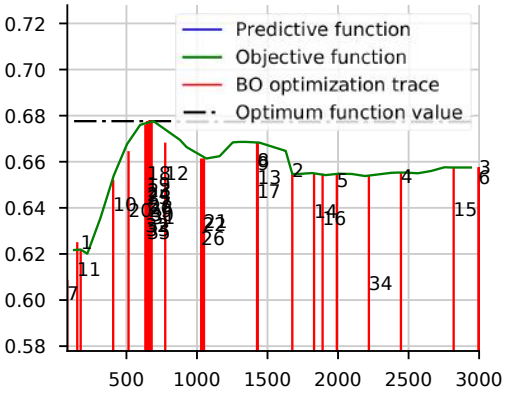
In this chapter, we have answered **RQ4** by studying the problem leveraging existing test collections to optimize retrieval systems. We propose the use of BO to jointly search and optimize over the hyperparameter space. Given the heterogeneous hyperparameters in retrieval systems we suggest the use of four covariance functions that can handle both continuous and categorical hyperparameters, the SE, HSE, Matérn1 and HMatérn1 covariance function. We analyze the effect of the different components of BO and reach the same conclusion as prior research on the topic [149] that it is the choice of the covariance functions that has the strongest effect on the performance of BO. To demonstrate the effectiveness and efficiency of BO to identify a good system configuration, we tested it on both the ad-hoc and the web test collections of TREC. In both collections we demonstrate that BO outperforms manual tuning, grid search and random search, both in terms of the retrieval effectiveness of the best configuration



(a) Round 1 predictive function

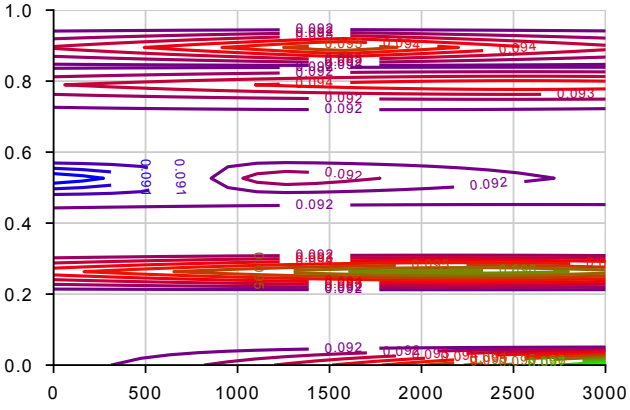


(b) Round 32 predictive function

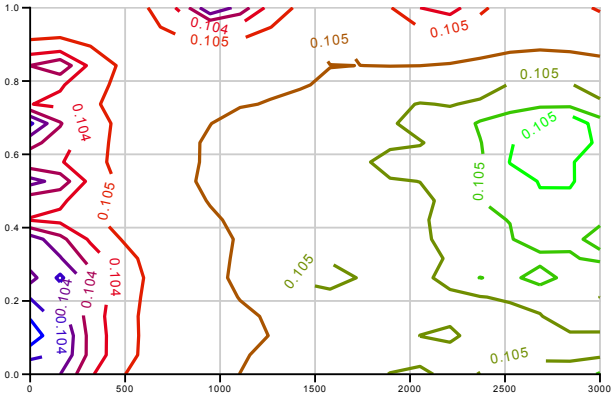


(c) Objective function

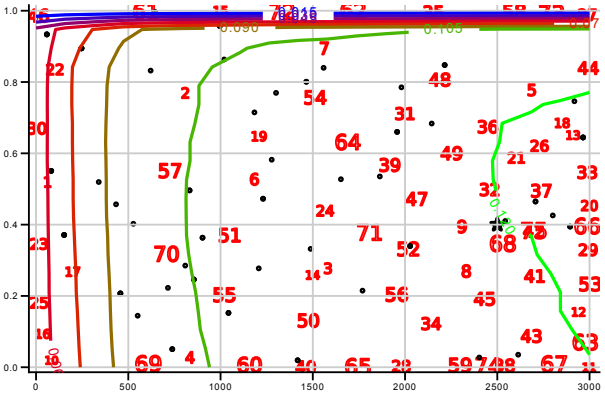
Figure 5.5: Zoomed-in result on TREC 6 + MRR. The first two panels are the predictive objective functions in round 1 and 32; the last panel is the real objective function. The red vertical line associated with a number denotes BO searches which point and in which round.



(a) Round 1 predictive function

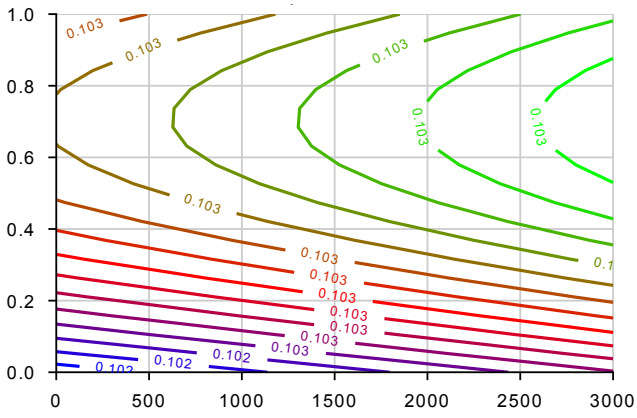


(b) Round 32 predictive function

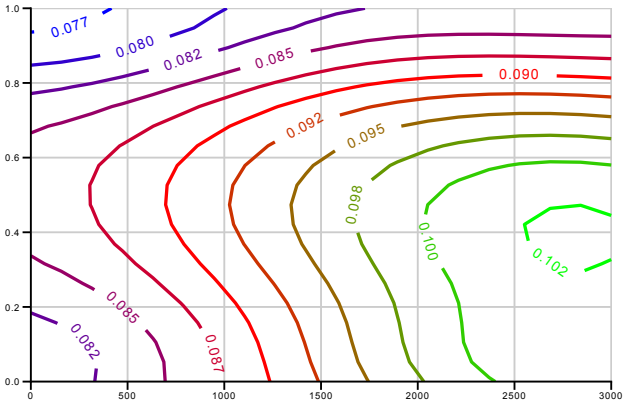


(c) Objective function

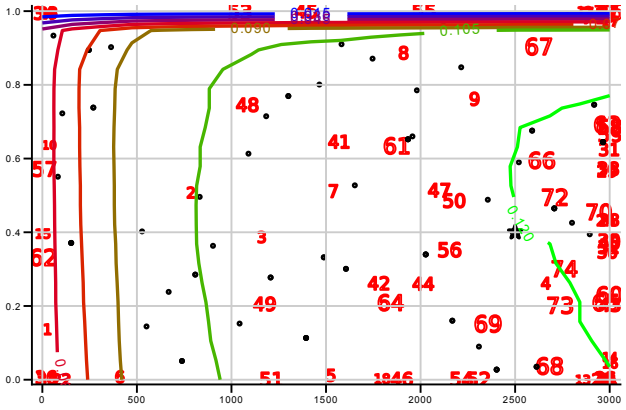
Figure 5.6: Matérn1 + MAP.



(a) Round 1 predictive function



(b) Round 32 predictive function



(c) Objective function

Figure 5.7: SE + MAP.

found, and in terms of efficiency in finding this configuration. We further examined the optimization behavior of BO in terms of exploitation and exploitation. We found that it spends more search budget near the global optimum or local optimums, and this optimization behavior is affected by covariance functions of different smoothness. One should note that for the GP the bounds for all dimensions of the search space are axis-aligned, i.e., the search space is a *hyper-rectangle* [144]. This is contradictory with the conditional structure of IR hyperparameters, which means that our instantiation of BO wastes time in searching in inactive dimensions. We leave the study of a surrogate model that can solve conditional hyperparameters as a future work.

Next, in Chapter 6, we conclude the thesis and suggest research directions for future work.

6 Conclusions

In the previous chapters, we have introduced methods to address the research questions raised in Chapter 1. In this chapter, we first revisit those research questions and summarize the main findings and implications of our research in Section 6.1. Then, in Section 6.2, we describe the main limitations of our work and possible future directions.

6.1 Main Findings

6.1.1 Document Selection through Active Sampling

We start with the problem of large-scale retrieval evaluation and ask:

RQ1 How can we effectively select documents in order to construct a test collection that allows for unbiased and low-variance effectiveness measures?

We devise a sample-based approach – active sampling – in Chapter 2. Our method consists of a module for sampling documents and a module for unbiased estimation of retrieval effectiveness. In the sampling module, we construct two distributions, one over retrieval systems that is updated at every round of relevance judgments, giving larger probabilities to better quality systems, and one over document ranks that is defined at the beginning of the sampling process and remains static throughout the experiment. Document samples are drawn from the joint probability distribution, and inclusion probabilities are computed at the end of the entire sampling process accounting for varying probabilities across sampling rounds. In the estimation module, we use the well-known Horvitz-Thompson estimator to estimate evaluation metrics for all systems.

We empirically verified the performance of the proposed method. We tested against state-of-the-art sample-based and active-selection methods over seven TREC collections, TREC 5–11, covering both ad-hoc and web search scenarios. The major answers are:

1. Compared to sample-based approaches, such as stratified sampling, out method indeed demonstrated lower variance, because the attention of our method is put on good quality runs with the hope of identifying more relevant documents and reduce the variability naturally introduced in the estimation of a measure due to sampling.
2. When compared against active-selection approaches, such as move-to-front, and

multi-armed bandits, our method has lower, near-zero bias, thus it can safely be used to evaluate new, novel runs that have not contributed to the generation of the test collection.
3. For sampling rates as low as 5% of the entire depth-100 pool, our proposed method outperforms all other methods regarding effectiveness and efficiency and leads to reusable test collections.

6.1.2 Automatic Thresholding of Document Selection

One of the assumptions of the Cranfield paradigm is that relevance assessments are complete. But complete relevance assessments require immense human effort. Therefore one needs to find the trade-off between high recall and low assessment cost. In this section we ask "when to stop selecting documents," or more specifically:

RQ2 How can we select documents for assessing relevance and stopping the selection, both in an effective manner to maximize recall and minimize assessment cost and in a transparent manner so that we know the number of residual relevant documents?

In Chapter 3, we propose a novel continuous active learning framework by jointly training a ranking model to rank documents, and conducting a "greedy" sampling to estimate the total number of relevant documents in the collection. Within the framework we propose to use the AP-Prior distribution as the sampling distribution, use the Horvitz-Thompson or Hansen-Hurwitz estimator to estimate the total number of relevant documents, and use a optimistic or conservative strategy to determine whether to stop the TAR process or not. We prove the unbiasedness of the proposed estimators under a with-replacement sampling design.

To examine the performance of the proposed thresholding method, we compared it against the knee, target, SCAL, SD-training and SD-sampling methods on various datasets including the CLEF Technology-Assisted Reviews in Empirical Medicine datasets [81, 83, 85], the TREC Total Recall datasets [60], and the TREC Legal datasets [41]. The experimental results demonstrate the following findings:
1. The proposed method performs better than the baselines. It combines the advantages of the continuous active learning approach with the advantages of sampling methods. It can effectively retrieve relevant documents similar to CAL but also provide a transparent, accurate, and effective stopping point.
2. Specifically, we recommend to use the AP-Prior distribution, the Horvitz-Thompson estimator together with the conservative stopping strategy. If efficiency is also considered, the Hansen-Hurwitz estimator is recommended to replace the Horvitz-Thompson estimator.

6.1.3 Aggregating Crowd Relevance Assessments

Another important component of constructing test collections is determining relevance labels. As crowdsourcing has arisen as a de facto solution, we ask:

RQ3 How can we effectively aggregate crowdsourcing labels in order to acquire high-quality labels in test collections?

In Chapter 4, we study the problem of relevance inference from noisy assessments. We propose a new annotation generation process modelled by a Bayesian generative model

called multi-annotator Gaussian process. This defines a novel data generation process. To be specific, our model employs a Gaussian process prior on the latent true labels of each task and learns the correlation among tasks; further, it employs Gaussian noise on each annotator and each task to learn task difficulty and annotator competence.

We designed three experiments to examine the performance of our label aggregation model. The experimental results demonstrate the following findings:

1. The proposed model is comparable with baselines. By comparing our method to simplified versions of it as baselines, we demonstrate that both the prior part and the likelihood part contribute to the improvement of the inferred label quality, while jointly modelling true labels of tasks and task difficulty can further help to construct a high quality test collection.
2. The annotation quality and redundancy both impact the performance of the proposed model, and annotation quality dominates the impact.

6.1.4 Optimizing Retrieval Systems

For the last work in this thesis, we explored the use of test collections to optimize retrieval systems and ask:

RQ4 How can we optimize the configuration of retrieval systems with regard to effectiveness measures on the basis of existing test collections?

In Chapter 5, we propose the use of BO to jointly search and optimize over the hyperparameter space of retrieval systems. To demonstrate the effectiveness and efficiency of BO to identify a good system configuration, we tested it on both the ad-hoc and the web test collections of TREC. The main answers are:

1. Given the heterogeneous hyperparameters in retrieval systems, we suggest the use of four covariance functions that can handle both continuous and categorical hyperparameters, the SE, HSE, Matérn1 and HMatérn1 covariance function;
2. We analyze the effect of the different components of BO and reach the same conclusion as prior research on the topic [149], that it is the choice of the covariance functions that has the strongest effect on the performance of BO.
3. BO outperforms manual tuning, grid search and random search, both in terms of the retrieval effectiveness of the best configuration found, and in terms of efficiency in finding this configuration.
4. We further examined the optimization behavior of BO in terms of exploitation and exploitation. We found that it spends more search budget near the global optimum or local optimums, and this optimization behavior is affected by covariance functions of different smoothness.

6.2 Future Work

One of the ultimate goals of the work in this thesis is to enable effective and efficient IR evaluation. In order to achieve the goal, it is worthwhile to re-examine the assumptions of the Cranfield evaluation paradigm. One of the assumptions is that the relevance assessments in the test collection are complete, even though this is not true in reality. Chapter 2 and 3 provide methods towards solving the issue. Another assumption is that the relevance of one document is independent of the relevance of other documents. But

this assumption can be relaxed. Chapter 4 addresses violations of this assumption by modelling the correlation of relevance between topics, documents and annotators who give the relevance assessments.

In this final part of the thesis, we recap the assumptions of the methods proposed in this thesis and discuss possible research directions for future work.

6.2.1 Document Selection through Active Sampling

The active sampling method in Chapter 2 consists of two modules: the sampling module and the estimation module. In the sampling module, we employ two distributions: one over retrieval systems, where the probabilities are weighted by the performance of system runs, and one over document ranks – the AP-prior distribution proposed by Aslam et al. [13] and Pavlu et al. [126]. One possible future direction is to explore different distributions over documents. The ideal case to make the distribution from which the documents are sampled, proportional to the distribution based on document relevancy, so that it ensures very low variance of the Horvitz-Thompson estimator [72] or the Hansen-Hurwitz estimator [158]. Existing work inspires us in choosing the sampling distribution. For example, Aslam et al. [10] and Yilmaz et al. [172] used a uniform distribution over the ranked document collection; Pavlu et al. [126] used stratified sampling to draw larger samples from the top ranks; Schnabel et al. [142] employed a weighted-importance sampling method on documents with the sampling distribution optimized for a comparative evaluation between runs. Another direction for future work is to sample documents without replacement. In this case, it is more efficient in terms of identifying relevant documents; on the other hand, it is also interesting to study how to efficiently calculate the Horvitz-Thompson estimator and the Hansen-Hurwitz estimator, and usually it is more difficult to calculate the inclusion probability for the Horvitz-Thompson estimator than the Hansen-Hurwitz estimator.

6.2.2 Automatic Thresholding of Document Selection

In Chapter 3 there are several directions that we have not touched on and that can be followed in future work. First, the thresholding method that we proposed is designed for small-scale document collections. It requires a relatively small list (tens of thousands) of documents instead of collections containing millions of documents so that the calculation of mean and variance of R is feasible. Currently, we adapted our method for large document collections by randomly splitting the documents, running our algorithm, and then concatenating the sampled documents for the final reviewing. However, this is not the best solution. More work can be done in the direction of training the ranking model globally to avoid sampling too many non-relevant documents. Second, like all other sampling methods, it inherently has a high variance problem for low-prevalence topics. In practice, where prevalence is low, one must resort to heuristic solutions such as snowball sampling, capture-recapture, and other techniques that are common in biostatistics and medicine. It is worth studying how to integrate these heuristic solutions in the proposed framework. Third, the performance of the ranking model can be further improved in order to produce a good ranked list of documents. Currently, only the document text instead of the topic text is employed for feature representation, the simple TF-IDF is used for document representation, the Logistic Regression model is used as the ranking model, and the ranking model is trained in a

topic-wise manner, i.e., a new ranking model is trained from scratch for each topic. In the future, a stronger ranking model can be trained by addressing these issues. Finally, the current model only considers the number of missing relevant documents to stop the TAR process. More factors should be taken into account such as the importance of the missing documents. For example, *risk of bias* and *quality*, two concepts from the systematic review domain indicating how reliable the results of the studies included in a systematic review are, can be leveraged to determine stopping the TAR process [68].

6.2.3 Aggregating Crowd Relevance Assessments

In Chapter 4 there are several directions that we have not touched on and that can be pursued in future work. The learning of workers' bias and tasks' bias. The mean of the Gaussian variable for workers in the proposed MAGP model could potentially model spammers or adversarial workers. The issue is similar for the tasks' bias. For the efficiency of learning the model parameters, we fix the mean of the Gaussian variable for workers or tasks to zero. In the future, it is interesting to adapt the inference algorithm to effectively learn the mean parameters.

6.2.4 Optimizing Retrieval Systems

In the Bayesian Optimization approach proposed in Chapter 5, we made an assumption that the search spaces are axis-aligned, i.e., the search space is a *hyper-rectangle* [144]. This is for the ease of using a Gaussian process regression model. However, this is contradictory with the conditional structure of the hyperparameters of the retrieval system, which means that our instantiation of BO wastes time in searching in inactive dimensions. One direction for future research is studying surrogate models that can solve the conditional-hyperparameter problem such as the tree-based Parzen estimator model [19].

In our work, in order to demonstrate the idea of using a machine learning approach to automatically configure a retrieval system on the basis of a specific test collection, we use the *Indri* system as an example of retrieval system and BO as an example of a machine learning approach. For future work, it is also interesting to apply other AutoML approaches [67] to more retrieval models.

Acronyms

Short	Full
AutoTAR	autonomous TAR (*see p.* 33)
BO	Bayesian optimization (*see p.* 6)
CAL	continuous active learning (*see p.* 32)
EI	expected improvement (*see p.* 110)
EM	expectation maximization (*see p.* 83)
EP	expectation propagation (*see p.* 85)
GP	Gaussian process (*see p.* 82)
GPMV	Gaussian process majority voting (*see p.* 97)
IR	information retrieval (*see p.* 1)
Latin	Latin hypercube sampling (*see p.* 110)
LK	likelihood (*see p.* 97)
MAB	multi-armed bandits (*see p.* 24)
MACE	multi-annotator competence estimation (*see p.* 97)
MAGP	multi-annotator Gaussian process (*see p.* 82)
MTF	move-to-front (*see p.* 24)
MV	majority voting (*see p.* 97)
NLP	natural learning processing (*see p.* 107)
PI	probability of improvement (*see p.* 110)
PRP	probability ranking principle (*see p.* 19)
Sobol	Sobol sequence based sampling (*see p.* 111)
Stratif	stratified sampling (*see p.* 24)
TAR	technology-assisted review (*see p.* 31)
TREC	text retrieval conference (*see p.* 2)
UCB	upper confidence bound (*see p.* 110)
Uniform	uniform random sampling (*see p.* 110)

Bibliography

[1] James Allan, Donna Harman, Evangelos Kanoulas, Dan Li, Christophe Van Gysel and Ellen M. Voorhees. 2017. TREC 2017 common core track overview. In *Proceedings of The Twenty-Sixth Text REtrieval Conference, TREC 2017, Gaithersburg, Maryland, USA, November 15-17, 2017* (NIST Special Publication). Volume 500-324. National Institute of Standards and Technology (NIST) (cited on pages 3, 11).

[2] Omar Alonso. 2019. *The Practice of Crowdsourcing*. *Synthesis Lectures on Information Concepts, Retrieval, and Services*. Morgan & Claypool Publishers (cited on pages 3, 84).

[3] Omar Alonso and Stefano Mizzaro. 2012. Using crowdsourcing for trec relevance assessment. *Information processing & management*, 48, 6, 1053–1066 (cited on pages 3, 81, 83).

[4] Omar Alonso, Daniel E Rose and Benjamin Stewart. 2008. Crowdsourcing for relevance evaluation. In *ACM SigIR Forum* number 2. Volume 42. ACM New York, NY, USA, 9–15 (cited on pages 3, 5, 82).

[5] Mauricio Álvarez, Lorenzo Rosasco, Neil Lawrence et al. 2012. Kernels for vector-valued functions: a review. *Foundations and Trends® in Machine Learning*, 4, 3, 195–266 (cited on page 85).

[6] Avi Arampatzis, Jaap Kamps and Stephen Robertson. 2009. Where to stop reading a ranked list?: threshold optimization using truncated score distributions. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 524–531 (cited on pages 35, 63).

[7] Timothy G. Armstrong, Alistair Moffat, William Webber and Justin Zobel. 2009. Improvements that don't add up: ad-hoc retrieval results since 1998. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (CIKM '09). ACM, Hong Kong, China, 601–610 (cited on pages 6, 105).

[8] Javed A Aslam and Mark Montague. 2001. Models for metasearch. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 276–284 (cited on page 93).

[9] Javed A Aslam and Virgil Pavlu. 2007. Query hardness estimation using jensen-shannon divergence among multiple scoring functions. In *European conference on information retrieval*. Springer, 198–209 (cited on page 38).

[10] Javed A Aslam, Virgil Pavlu and Emine Yilmaz. 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*. ACM, 541–548 (cited on pages 4, 16, 23, 24, 38, 130).

[11] Javed A. Aslam, Virgiliu Pavlu and Robert Savell. 2003. A unified model for metasearch, pooling, and system evaluation. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management* (CIKM '03). ACM, New Orleans, LA, USA, 484–491 (cited on pages 4, 16, 23).

[12] Javed A Aslam, Virgiliu Pavlu and Robert Savell. 2003. A unified model for metasearch, pooling, and system evaluation. In *Proceedings of the twelfth international conference on Information and knowledge management*. ACM, 484–491 (cited on page 38).

[13] Javed A Aslam, Virgiliu Pavlu and Emine Yilmaz. 2005. Measure-based metasearch. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 571–572 (cited on pages 19, 20, 38, 39, 130).

[14] Yoram Bachrach, Thore Graepel, Tom Minka and John Guiver. 2012. How to grade a test without knowing the answers—a bayesian graphical model for adaptive crowdsourcing and aptitude testing. *arXiv preprint arXiv:1206.6386* (cited on pages 84, 92).

[15] Mossaab Bagdouri, William Webber, David D Lewis and Douglas W Oard. 2013. Towards minimizing the annotation cost of certified text classification. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 989–998 (cited on page 35).

[16] Richard E Bellman. 2015. *Adaptive control processes: a guided tour*. Princeton university press (cited on page 106).

[17] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13, Feb, 281–305 (cited on pages 106, 116).

[18] James Bergstra, Daniel Yamins and David D Cox. 2013. Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. *ICML (1)*, 28, 115–123 (cited on pages 106, 107).

[19] James S Bergstra, Rémi Bardenet, Yoshua Bengio and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, 2546–2554 (cited on pages 106–108, 131).

[20] Anthony Bigot, Sébastien Déjean and Josiane Mothe. 2015. Learning to choose the best system configuration in information retrieval: the case of repeated queries. *Journal of Universal Computer Science*, 21, 13, 1726–1745 (cited on page 107).

[21] Christopher M Bishop. 2006. Pattern recognition. *Machine Learning*, 128, 1–58 (cited on page 23).

[22] Daren C Brabham. 2008. Crowdsourcing as a model for problem solving: an introduction and cases. *Convergence*, 14, 1, 75–90 (cited on page 3).

[23] Chris Buckley, Darrin Dimmick, Ian Soboroff and Ellen Voorhees. 2007. Bias and the limits of pooling for large collections. *Inf. Retr.*, 10, 6, (December 2007), 491–508 (cited on pages 2, 4, 15).

[24] Chris Buckley, Matthew Lease and Mark D. Smucker. 2010. Overview of the TREC 2010 Relevance Feedback Track (Notebook). In *The Nineteenth Text Retrieval Conference (TREC) Notebook* (cited on pages 3, 94).

[25] Chris Buckley and Ellen M. Voorhees. 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). ACM, Sheffield, United Kingdom, 25–32 (cited on pages 2, 15).

[26] Russel E Caflisch, William J Morokoff and Art B Owen. 1997. *Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension*. Department of Mathematics, University of California, Los Angeles (cited on page 106).

[27] Ben Carterette, Virgiliu Pavlu, Evangelos Kanoulas, Javed A. Aslam and James Allan. 2009. If I had a million queries. In *Advances in Information Retrieval, 31th European Conference on IR Research, ECIR 2009, Toulouse, France, April 6-9, 2009. Proceedings* (Lecture Notes in Computer Science). Mohand Boughanem, Catherine Berrut, Josiane Mothe and Chantal Soulé-Dupuy, editors. Volume 5478. Springer, 288–300 (cited on page 24).

[28] Suming J Chen, Xuanhui Wang, Zhen Qin and Donald Metzler. 2020. Parameter tuning in personal search systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 97–105 (cited on page 6).

[29] Charles L Clarke, Nick Craswell and Ian Soboroff. 2009. Overview of the trec 2009 web track. Technical report. WATERLOO UNIV (ONTARIO) (cited on page 93).

[30] Charles L. A. Clarke, Nick Craswell, Ian Soboroff and Gordon V. Cormack. 2010. Overview of the TREC 2010 web track. In *Proceedings of The Nineteenth Text REtrieval Conference, TREC 2010, Gaithersburg, Maryland, USA, November 16-19, 2010* (NIST Special Publication). Ellen M. Voorhees and Lori P. Buckland, editors. Volume 500-294. National Institute of Standards and Technology (NIST) (cited on page 93).

[31] Cyril Cleverdon. 1967. The cranfield tests on index language devices. In *Aslib proceedings* number 6. Volume 19. MCB UP Ltd, 173–194 (cited on pages 4, 15).

[32] Cyril W Cleverdon. 1967. The cranfield tests on index language devices. In *Aslib Proceedings*, 19:173–192 (cited on pages 1, 81).

[33] Cyril W Cleverdon and E Michael Keen. 1966. Factors determining the performance of indexing systems. *Aslib Cranfield Research Project*, 2 (cited on page 1).

[34] Cyril W Cleverdon, Jack Mills and E Michael Keen. 1966. Factors determining the performance of indexing systems. *Cranfield: College of Aeronautics*, 1 (cited on page 1).

[35] Gordon V. Cormack and Maura R. Grossman. 2015. Autonomy and reliability of continuous active learning for technology-assisted review. *CoRR*, abs/1504.06868. arXiv: 1504.06868 (cited on pages 5, 32–34, 36–38, 58).

[36] Gordon V. Cormack and Maura R. Grossman. 2018. Beyond pooling. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (SIGIR '18). ACM, Ann Arbor, MI, USA, 1169–1172 (cited on pages 5, 32).

[37] Gordon V. Cormack and Maura R. Grossman. 2016. Engineering quality and reliability in technology-assisted review. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '16). ACM, Pisa, Italy, 75–84 (cited on pages 5, 7, 8, 32, 34, 36, 57–60, 66, 70).

[38] Gordon V Cormack and Maura R Grossman. 2014. Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 153–162 (cited on pages 5, 32, 33).

[39] Gordon V. Cormack and Maura R. Grossman. 2016. Scalability of continuous active learning for reliable high-recall text classification. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (CIKM '16). ACM, Indianapolis, Indiana, USA, 1039–1048 (cited on pages 5, 7, 8, 32, 34, 36, 58, 60–63, 66).

[40] Gordon V Cormack and Maura R Grossman. 2017. Technology-assisted review in empirical medicine: waterloo participation in clef ehealth 2017. In *CLEF (Working Notes)* (cited on pages 58, 59).

[41] Gordon V Cormack, Maura R Grossman, Bruce Hedin and Douglas W Oard. 2010. Overview of the trec 2010 legal track. In *Proc. 19th Text REtrieval Conference*. Volume 1 (cited on pages 4, 31, 33, 54, 78, 128).

[42] Gordon V. Cormack, Christopher R. Palmer and Charles L. A. Clarke. 1998. Efficient construction of large test collections. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '98). ACM, Melbourne, Australia, 282–289 (cited on pages 4, 31).

[43] Gordon V Cormack, Christopher R Palmer and Charles LA Clarke. 1998. Efficient construction of large test collections. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 282–289 (cited on pages 4, 7, 16, 24).

[44] Jeffrey Dalton, Chenyan Xiong and Jamie Callan. 2020. TREC cast 2019: the conversational assistance track overview. *CoRR*, abs/2003.13624. arXiv: 2003.13624 (cited on page 3).

[45] Martin Davtyan, Carsten Eickhoff and Thomas Hofmann. 2015. Exploiting document content for efficient aggregation of crowdsourcing votes. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 783–790 (cited on page 84).

[46] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, 20–28 (cited on pages 84, 92).

[47] Jeffrey Dean. 2020. 1.1 the deep learning revolution and its implications for computer architecture and chip design. In *2020 IEEE International Solid- State Circuits Conference, ISSCC 2020, San Francisco, CA, USA, February 16-20, 2020*. IEEE, 8–14 (cited on page 1).

[48] Romain Deveaud, Josiane Mothe and Jian-Yun Nie. 2016. Learning to rank system configurations. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (CIKM '16). ACM, Indianapolis, Indiana, USA, 2001–2004 (cited on page 107).

[49] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 2018. Bert: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (cited on page 93).

[50] Giorgio Maria Di Nunzio. 2018. A study of an automatic stopping strategy for technologically assisted medical reviews. In *European Conference on Information Retrieval*. Springer, 672–677 (cited on pages 5, 32, 35).

[51] Laurence Charles Ward Dixon and Giorgio Philip Szegö. 1978. *Towards global optimisation 2*. North-Holland Amsterdam. Chapter The Application of Bayesian Methods for Seeking the Extremum (cited on page 110).

[52] Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos and Kevin Leyton-Brown. 2013. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, 1–5 (cited on pages 6, 106).

[53] Carsten Eickhoff. 2018. Cognitive biases in crowdsourcing. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (WSDM '18). ACM, Marina Del Rey, CA, USA, 162–170 (cited on page 5).

[54] Marco Ferrante, Nicola Ferro and Eleonora Losiouk. 2019. Stochastic relevance for crowdsourcing. In *European Conference on Information Retrieval*. Springer, 755–762 (cited on page 84).

[55] Marco Ferrante, Nicola Ferro and Maria Maistro. 2017. Aware: exploiting evaluation measures to combine multiple assessors. *ACM Transactions on Information Systems (TOIS)*, 36, 2, 20 (cited on page 85).

[56] Nicola Ferro and Gianmaria Silvello. 2016. A general linear mixed models approach to study system component effects. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '16). ACM, Pisa, Italy, 25–34 (cited on pages 105, 114).

[57] Nicola Ferro and Gianmaria Silvello. 2018. Toward an anatomy of ir system component performances. *Journal of the Association for Information Science and Technology*, 69, 2, 187–200 (cited on page 6).

[58] Parantapa Goswami and Eric Gaussier. 2013. Estimation of the collection parameter of information models for ir. In *Proceedings of the 35th European Conference on Advances in Information Retrieval* (ECIR'13). Springer-Verlag, Moscow, Russia, 459–470 (cited on pages 3, 6, 105, 107).

[59] Perry Groot, Adriana Birlutiu and Tom Heskes. 2011. Learning from multiple annotators with gaussian processes. In *International Conference on Artificial Neural Networks*. Springer, 159–164 (cited on page 85).

[60] Maura R Grossman, Gordon V. Cormack and Adam Roegiest. 2016. Trec 2016 total recall track overview. In *TREC* (cited on pages 4, 31, 33, 44, 53, 57, 78, 128).

[61] Christophe Van Gysel, Dan Li and Evangelos Kanoulas. 2017. ILPS at TREC 2017 common core track. In *Proceedings of The Twenty-Sixth Text REtrieval Conference, TREC 2017, Gaithersburg, Maryland, USA, November 15-17, 2017* (NIST Special Publication). Volume 500-324. National Institute of Standards and Technology (NIST) (cited on page 11).

[62] Lei Han, Eddy Maddalena, Alessandro Checco, Cristina Sarasua, Ujwal Gadiraju, Kevin Roitero and Gianluca Demartini. 2020. Crowd worker strategies in relevance judgment tasks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 241–249 (cited on pages 3, 81, 83).

[63] Lei Han, Kevin Roitero, Ujwal Gadiraju, Cristina Sarasua, Alessandro Checco, Eddy Maddalena and Gianluca Demartini. 2019. All those wasted hours: on task abandonment in crowdsourcing. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 321–329 (cited on page 82).

[64] Ben HE and Iadh Ounis. 2003. A study of parameter tuning for term frequency normalization. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management* (CIKM '03). ACM, New Orleans, LA, USA, 10–16 (cited on pages 3, 6, 105, 107).

[65] Ben He and Iadh Ounis. 2007. On setting the hyper-parameters of term frequency normalization for information retrieval. *ACM Trans. Inf. Syst.*, 25, 3, (July 2007) (cited on pages 3, 6, 105, 107).

[66] Ben He and Iadh Ounis. 2007. Parameter sensitivity in the probabilistic model for ad-hoc retrieval. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management* (CIKM '07). ACM, Lisbon, Portugal, 263–272 (cited on pages 3, 6, 105, 107).

[67] Xin He, Kaiyong Zhao and Xiaowen Chu. 2019. Automl: A survey of the state-of-the-art. *CoRR*, abs/1908.00709. arXiv: 1908.00709 (cited on page 131).

[68] Julian PT Higgins and Douglas G Altman. 2008. Assessing risk of bias in included studies. *Cochrane handbook for systematic reviews of interventions: Cochrane book series*, 187–241 (cited on pages 76, 79, 131).

[69] Matthew W Hoffman and Bobak Shahriari. 2014. Modular mechanisms for bayesian optimization. In *NIPS Workshop on Bayesian Optimization*. Citeseer (cited on pages 111, 114).

[70] Katja Hofmann, Shimon Whiteson and Maarten de Rijke. 2011. A probabilistic method for inferring preferences from clicks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management* (CIKM '11). ACM, Glasgow, Scotland, UK, 249–258 (cited on page 19).

[71] Noah Hollmann and Carsten Eickhoff. 2017. Ranking and feedback-based stopping for recall-centric document retrieval. In *CLEF (Working Notes)* (cited on pages 7, 8, 35, 58, 63, 64).

[72] Daniel G Horvitz and Donovan J Thompson. 1952. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47, 260, 663–685 (cited on pages 21, 41, 130).

[73] Mehdi Hosseini, Ingemar J Cox, Nataša Milić-Frayling, Gabriella Kazai and Vishwa Vinay. 2012. On aggregating labels from multiple crowd workers to infer relevance of documents. In *European Conference on Information Retrieval*. Springer, 182–194 (cited on page 84).

[74] Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani and Eduard Hovy. 2013. Learning whom to trust with mace. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1120–1130 (cited on pages 84, 85, 92, 97, 99).

[75] Frank Hutter, Holger H Hoos and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*. Springer, 507–523 (cited on pages 106, 107, 112).

[76] Frank Hutter and Michael A Osborne. 2013. A kernel for hierarchical parameter spaces. *arXiv preprint arXiv:1310.5738* (cited on page 107).

[77] Oana Inel, Giannis Haralabopoulos, Dan Li, Christophe Van Gysel, Zoltán Szlávik, Elena Simperl, Evangelos Kanoulas and Lora Aroyo. 2018. Studying topical relevance with evidence-based

crowdsourcing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. ACM, 1253–1262 (cited on page 10).

[78] Ayush Jain, Akash Das Sarma, Aditya Parameswaran and Jennifer Widom. 2017. Understanding workers, developing effective tasks, and enhancing marketplace dynamics: a study of a large crowdsourcing marketplace. *Proceedings of the VLDB Endowment*, 10, 7, 829–840 (cited on pages 81, 85, 96).

[79] Karen Sparck Jones. 1981. The cranfield tests (cited on page 1).

[80] Evangelos Kanoulas. 2015. A short survey on online and offline methods for search quality evaluation. In *Information Retrieval - 9th Russian Summer School, RuSSIR 2015, Saint Petersburg, Russia, August 24-28, 2015, Revised Selected Papers* (Communications in Computer and Information Science). Pavel Braslavski, Ilya Markov, Panos M. Pardalos, Yana Volkovich, Dmitry I. Ignatov, Sergei Koltsov and Olessia Koltsova, editors. Volume 573. Springer, 38–87 (cited on pages 2, 15).

[81] Evangelos Kanoulas, Dan Li, Leif Azzopardi and Rene Spijker. 2017. Clef 2017 technologically assisted reviews in empirical medicine overview. In *CEUR Workshop Proceedings*. Volume 1866, 1–29 (cited on pages 4, 31, 33, 45, 47, 48, 53, 78, 128).

[82] Evangelos Kanoulas, Dan Li, Leif Azzopardi and René Spijker. 2017. CLEF 2017 technologically assisted reviews in empirical medicine overview. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017* (CEUR Workshop Proceedings). Volume 1866. CEUR-WS.org (cited on pages 9, 57).

[83] Evangelos Kanoulas, Dan Li, Leif Azzopardi and Rene Spijker. 2018. Clef 2018 technologically assisted reviews in empirical medicine overview. In *CEUR Workshop Proceedings* (cited on pages 4, 31, 33, 45, 48, 53, 78, 128).

[84] Evangelos Kanoulas, Dan Li, Leif Azzopardi and René Spijker. 2018. CLEF 2018 technologically assisted reviews in empirical medicine overview. In *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018* (CEUR Workshop Proceedings). Volume 2125. CEUR-WS.org (cited on pages 9, 57).

[85] Evangelos Kanoulas, Dan Li, Leif Azzopardi and Rene Spijker. [n. d.] Clef 2019 technology assisted reviews in empirical medicine overview. In (cited on pages 4, 31, 33, 45, 48, 53, 78, 128).

[86] Evangelos Kanoulas, Dan Li, Leif Azzopardi and René Spijker. 2019. CLEF 2019 technology assisted reviews in empirical medicine overview. In *Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum, Lugano, Switzerland, September 9-12, 2019* (CEUR Workshop Proceedings). Volume 2380. CEUR-WS.org (cited on pages 10, 57).

[87] Evangelos Kanoulas, Virgil Pavlu, Keshi Dai and Javed A Aslam. 2009. Modeling the score distributions of relevant and non-relevant documents. In *Conference on the Theory of Information Retrieval*. Springer, 152–163 (cited on page 35).

[88] Gabriella Kazai, Jaap Kamps and Natasa Milic-Frayling. 2013. An analysis of human factors and label accuracy in crowdsourcing relevance judgments. *Information retrieval*, 16, 2, 138–178 (cited on pages 3, 81, 82).

[89] Liadh Kelly, Hanna Suominen, Lorraine Goeuriot, Mariana L. Neves, Evangelos Kanoulas, Dan Li, Leif Azzopardi, René Spijker, Guido Zuccon, Harrisen Scells and João R. M. Palotti. 2019. Overview of the CLEF ehealth evaluation lab 2019. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 10th International Conference of the CLEF Association, CLEF 2019, Lugano, Switzerland, September 9-12, 2019, Proceedings* (Lecture Notes in Computer Science). Volume 11696. Springer, 322–339 (cited on page 11).

[90] Sadegh Kharazmi, Falk Scholer, David Vallet and Mark Sanderson. 2016. Examining additivity and weak baselines. *ACM Trans. Inf. Syst.*, 34, 4, (June 2016), 23:1–23:18 (cited on pages 6, 105).

[91] Hyun-Chul Kim and Zoubin Ghahramani. 2012. Bayesian classifier combination. In *Artificial Intelligence and Statistics*, 619–627 (cited on page 84).

[92] Matthew Lease. 2011. On quality control and machine learning in crowdsourcing. In *Proceedings of the 11th AAAI Conference on Human Computation* (AAAIWS'11-11). AAAI Press, 97–102 (cited on pages 5, 81).

[93] Dan Li and Evangelos Kanoulas. 2020. A multi-annotator gaussian process model for relevance inference from noisy annotations. *The Web Conference 2021*. Under submission (cited on page 10).

[94] Dan Li and Evangelos Kanoulas. 2017. Active sampling for large-scale information retrieval evaluation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 49–58 (cited on pages 4, 31, 38).

[95] Dan Li and Evangelos Kanoulas. 2017. Active sampling for large-scale information retrieval evaluation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (CIKM '17). Association for Computing Machinery, Singapore, Singapore, 49–58 (cited on page 9).

[96] Dan Li and Evangelos Kanoulas. 2019. Automatic thresholding by sampling documents and estimating recall. In *Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum, Lugano, Switzerland, September 9-12, 2019* (CEUR Workshop Proceedings). Volume 2380. CEUR-WS.org (cited on page 11).

[97] Dan Li and Evangelos Kanoulas. 2018. Bayesian optimization for optimizing retrieval systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (WSDM '18). Association for Computing Machinery, Marina Del Rey, CA, USA, 360–368 (cited on page 10).

[98] Dan Li and Evangelos Kanoulas. 2020. When to stop reviewing in technology-assisted reviews. *The ACM Transactions on Information Systems*. Accepted, to appear (cited on page 9).

[99] Dan Li, Panagiotis Zafeiriadis and Evangelos Kanoulas. 2020. APS: an active PubMed search system for technology assisted review. In *The 43st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2020, Xi'an, China, July 25-30, 2020*. ACM (cited on page 10).

[100] Guoliang Li, Jiannan Wang, Yudian Zheng and Michael J Franklin. 2016. Crowdsourced data management: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 28, 9, 2296–2319 (cited on pages 82, 84, 95).

[101] Hang Li. 2014. *Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition*. *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers (cited on page 23).

[102] Yuan Li. 2019. *Probabilistic models for aggregating crowdsourced annotations*. PhD thesis (cited on page 85).

[103] Yuan Li, Benjamin Rubinstein and Trevor Cohn. 2019. Exploiting worker correlation for label aggregation in crowdsourcing. In *International Conference on Machine Learning*, 3886–3895 (cited on pages 84, 92).

[104] 2016. *The curious incidence of bias corrections in the pool*. *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings*. Springer International Publishing, Cham, 267–279 (cited on page 16).

[105] Aldo Lipani, Mihai Lupu, Evangelos Kanoulas and Allan Hanbury. 2016. The solitude of relevant documents in the pool. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (CIKM '16). ACM, Indianapolis, Indiana, USA, 1989–1992 (cited on page 16).

[106] Aldo Lipani, Mihai Lupu, Joao Palotti, Guido Zuccon and Allan Hanbury. 2017. Fixed budget pooling strategies based on fusion methods. In *Proc. of SAC* (cited on pages 4, 16).

[107] Aldo Lipani, Guido Zuccon, Mihai Lupu, Bevan Koopman and Allan Hanbury. 2016. The impact of fixed-cost pooling strategies on test collection bias. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval* (ICTIR '16). ACM, Newark, Delaware, USA, 105–108 (cited on pages 2, 4, 15, 16).

[108] Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong and Hang Li. 2007. Letor: benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*. Volume 310. ACM Amsterdam, The Netherlands (cited on page 92).

[109] David E Losada, Javier Parapar and Álvaro Barreiro. 2016. Feeling lucky?: multi-armed bandits for ordering judgements in pooling-based evaluation. In *proceedings of the 31st annual ACM symposium on applied computing*. ACM, 1027–1034 (cited on pages 4, 7, 16, 24).

[110] David E Losada, Javier Parapar and Alvaro Barreiro. 2019. When to stop making relevance judgments? a study of stopping methods for building information retrieval test collections. *Journal of the Association for Information Science and Technology*, 70, 1, 49–60 (cited on pages 5, 32, 35).

[111] Yuanhua Lv and ChengXiang Zhai. 2009. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (CIKM '09). ACM, Hong Kong, China, 1895–1898 (cited on pages 3, 6, 105, 107).

[112] Eddy Maddalena, Marco Basaldella, Dario De Nart, Dante Degl'Innocenti, Stefano Mizzaro and Gianluca Demartini. 2016. Crowdsourcing relevance assessments: the unexpected benefits of lim-

iting the time to judge. In *Fourth AAAI conference on human computation and crowdsourcing* (cited on page 83).

[113] Eddy Maddalena, Stefano Mizzaro, Falk Scholer and Andrew Turpin. 2017. On crowdsourcing relevance magnitudes for information retrieval evaluation. *ACM Trans. Inf. Syst.*, 35, 3, (January 2017), 19:1–19:32 (cited on page 31).

[114] Eddy Maddalena, Kevin Roitero, Gianluca Demartini and Stefano Mizzaro. 2017. Considering assessor agreement in ir evaluation. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, 75–82 (cited on pages 3, 82).

[115] Christopher D Manning, Prabhakar Raghavan and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge university press (cited on page 82).

[116] Iain J Marshall and Byron C Wallace. 2019. Toward systematic review automation: a practical guide to using machine learning tools in research synthesis. *Systematic reviews*, 8, 1, 163 (cited on page 44).

[117] De G Matthews, G Alexander, Mark Van Der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani and James Hensman. 2017. Gpflow: a gaussian process library using tensorflow. *The Journal of Machine Learning Research*, 18, 1, 1299–1304 (cited on page 94).

[118] Tyler McDonnell, Matthew Lease, Mucahid Kutlu and Tamer Elsayed. 2016. Why is that relevant? collecting annotator rationales for relevance judgments. In *Fourth AAAI Conference on Human Computation and Crowdsourcing* (cited on pages 3, 83).

[119] Jonas Mockus. 1994. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4, 4, 347–365 (cited on page 108).

[120] Pablo Morales-Álvarez, Pablo Ruiz, Raúl Santos-Rodríguez, Rafael Molina and Aggelos K Katsaggelos. 2019. Scalable and efficient learning from crowds with gaussian processes. *Information Fusion*, 52, 110–127 (cited on pages 85, 86).

[121] Nasser M Nasrabadi. 2007. Pattern recognition and machine learning. *Journal of electronic imaging*, 16, 4, 049901 (cited on page 91).

[122] Radford M Neal and Geoffrey E Hinton. 1998. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*. Springer, 355–368 (cited on page 91).

[123] Douglas W. Oard, Fabrizio Sebastiani and Jyothi K. Vinjumur. 2018. Jointly minimizing the expected costs of review for responsiveness and privilege in e-discovery. *ACM Trans. Inf. Syst.*, 37, 1, (November 2018), 11:1–11:35 (cited on page 31).

[124] Alison O'Mara-Eves, James Thomas, John McNaught, Makoto Miwa and Sophia Ananiadou. 2015. Using text mining for study identification in systematic reviews: a systematic review of current approaches. *Systematic reviews*, 4, 1, 5 (cited on pages 31, 44).

[125] Manfred Opper and Cédric Archambeau. 2009. The variational gaussian approximation revisited. *Neural computation*, 21, 3, 786–792 (cited on pages 91, 92).

[126] V Pavlu and J Aslam. 2007. A practical sampling strategy for efficient retrieval evaluation. Technical report. Technical report, Northeastern University (cited on pages 4, 7, 16, 19, 20, 23, 24, 38, 39, 130).

[127] Des Raj. 1964. A note on the variance of the ratio estimate. *Journal of the American Statistical Association*, 59, 307, 895–898 (cited on page 22).

[128] Carl Edward Rasmussen. 2004. Gaussian processes in machine learning. In *Advanced lectures on machine learning*. Springer, 63–71 (cited on pages 85, 87).

[129] Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press (cited on page 111).

[130] Alexander Jason Ratner. 2019. *Accelerating Machine Learning with Training Data Management*. PhD thesis. Stanford University (cited on page 1).

[131] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research*, 11, Apr, 1297–1322 (cited on pages 84, 92).

[132] S. E. Robertson. 1997. Readings in information retrieval. In Karen Sparck Jones and Peter Willett, editors. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. Chapter The Probability Ranking Principle in IR, 281–286 (cited on pages 19, 38).

[133] Stephen Robertson, Evangelos Kanoulas and Emine Yilmaz. 2013. Modelling score distributions without actual scores. In *Proceedings of the 2013 Conference on the Theory of Information Retrieval*. ACM, 20 (cited on page 35).

[134] Filipe Rodrigues, Francisco Pereira and Bernardete Ribeiro. 2014. Gaussian process classification and active learning with multiple annotators. In *International Conference on Machine Learning*, 433–441 (cited on pages 85, 86).

[135] Cormack Gordon V Grossman Maura R Roegiest Adam and Clarke Charles L A Cormack. 2015. Trec 2015 total recall track overview. In *TREC* (cited on pages 4, 31, 44, 57).

[136] François Rousseau and Michalis Vazirgiannis. 2013. Composition of tf normalizations: new insights on scoring functions for ad hoc ir. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '13). ACM, Dublin, Ireland, 917–920 (cited on pages 3, 6, 105, 107).

[137] Pablo Ruiz, Pablo Morales-Álvarez, Rafael Molina and Aggelos K Katsaggelos. 2019. Learning from crowds with variational gaussian processes. *Pattern Recognition*, 88, 298–311 (cited on pages 85, 86).

[138] G. Salton and M. E. Lesk. 1965. The smart automatic document retrieval systems?an illustration. *Commun. ACM*, 8, 6, (June 1965), 391?398 (cited on page 2).

[139] Mark Sanderson. 2010. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval*, 4, 4, 247–375 (cited on pages 2, 15).

[140] V Satopaa, J Albrecht, D Irwin and B Raghavan. 2011. Finding a 'kneedle' in a haystack: detecting knee points in system behavior. 166–171. In *31-st International Conference on Distributed Computing Systems. See: http://www1. icsi. berkeley. edu/barath/papers/kneedle-simplex11. pdf* (cited on pages 34, 58).

[141] Harrisen Scells, Guido Zuccon, Bevan Koopman, Anthony Deacon, Shlomo Geva and Leif Azzopardi. 2017. A test collection for evaluating retrieval of studies for inclusion in systematic reviews. In *To appear in Proceedings of the 40th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM (cited on page 48).

[142] Tobias Schnabel, Adith Swaminathan, Peter I. Frazier and Thorsten Joachims. 2016. Unbiased comparative evaluation of ranking functions. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval* (ICTIR '16). ACM, Newark, Delaware, USA, 109–118 (cited on pages 4, 16, 20, 130).

[143] Jangwon Seo and W. Bruce Croft. 2010. Unsupervised estimation of dirichlet smoothing parameters. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '10). ACM, Geneva, Switzerland, 759–760 (cited on pages 3, 6, 105, 107).

[144] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams and Nando de Freitas. 2016. Taking the human out of the loop: a review of bayesian optimization. *Proceedings of the IEEE*, 104, 1, 148–175 (cited on pages 106–108, 110, 126, 131).

[145] Ian Shemilt, Nada Khan, Sophie Park and James Thomas. 2016. Use of cost-effectiveness analysis to compare the efficiency of study identification methods in systematic reviews. *Systematic Reviews*, 5, 1, 140 (cited on page 45).

[146] Mark D. Smucker, Gabriella Kazai and Matthew Lease. 2012. Overview of the TREC 2012 crowdsourcing track. In *Proceedings of The Twenty-First Text REtrieval Conference, TREC 2012, Gaithersburg, Maryland, USA, November 6-9, 2012* (NIST Special Publication). Ellen M. Voorhees and Lori P. Buckland, editors. Volume 500-298. National Institute of Standards and Technology (NIST) (cited on pages 3, 83).

[147] Mark D. Smucker, Gabriella Kazai and Matthew Lease. 2012. Overview of the TREC 2012 crowdsourcing track. In *Proceedings of The Twenty-First Text REtrieval Conference, TREC 2012, Gaithersburg, Maryland, USA, November 6-9, 2012* (NIST Special Publication). Ellen M. Voorhees and Lori P. Buckland, editors. Volume 500-298. National Institute of Standards and Technology (NIST) (cited on page 83).

[148] Mark D. Smucker, Gabriella Kazai and Matthew Lease. 2013. Overview of the TREC 2013 crowdsourcing track. In *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013* (NIST Special Publication). Ellen M. Voorhees, editor. Volume 500-302. National Institute of Standards and Technology (NIST) (cited on page 83).

[149] Jasper Snoek, Hugo Larochelle and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2951–2959 (cited on pages 106, 107, 111, 117, 122, 129).

[150] Ian M Soboroff, Nick Craswell, Charles L Clarke and Gordon Cormack. 2011. Overview of the trec 2011 web track. Technical report (cited on page 93).

[151] K. Sparck Jones and C.J. van Rijsbergen. 1975. Report on the need for and provision of an 'ideal' information retrieval test collection. Technical report. Computer Laboratory, Cambridge University (cited on pages 4, 15).

[152] Trevor Strohman, Donald Metzler, Howard Turtle and W. Bruce Croft. 2005. Indri: a language-model based search engine for complex queries. Technical report. in Proceedings of the International Conference on Intelligent Analysis (cited on page 113).

[153] Hanna Suominen, Liadh Kelly, Lorraine Goeuriot, Aurélie Névéol, Lionel Ramadier, Aude Robert, Evangelos Kanoulas, René Spijker, Leif Azzopardi, Dan Li, Jimmy, João R. M. Palotti and Guido Zuccon. 2018. Overview of the CLEF ehealth evaluation lab 2018. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 9th International Conference of the CLEF Association, CLEF 2018, Avignon, France, September 10-14, 2018, Proceedings* (Lecture Notes in Computer Science). Volume 11018. Springer, 286–301 (cited on page 11).

[154] Hanna Suominen, Liadh Kelly, Lorraine Goeuriot, Aurélie Névéol, Lionel Ramadier, Aude Robert, Evangelos Kanoulas, Rene Spijker, Leif Azzopardi, Dan Li et al. 2018. Overview of the clef ehealth evaluation lab 2018. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 286–301 (cited on page 53).

[155] Kevin Swersky, David Duvenaud, Jasper Snoek, Frank Hutter and Michael A Osborne. 2014. Raiders of the lost architecture: kernels for bayesian optimization in conditional parameter spaces. *arXiv preprint arXiv:1409.4011* (cited on page 107).

[156] Kevin Swersky, Jasper Snoek and Ryan P Adams. 2013. Multi-task bayesian optimization. In *Advances in neural information processing systems*, 2004–2012 (cited on pages 106, 107).

[157] Michael Taylor, Hugo Zaragoza, Nick Craswell, Stephen Robertson and Chris Burges. 2006. Optimisation methods for ranking functions with multiple parameters. In *Proceedings of the 15th ACM international conference on Information and knowledge management*. ACM, 585–593 (cited on page 107).

[158] Steven K. Thompson. 2012. *Sampling*. (3rd edition). John Wiley & Sons, Inc., Hoboken, New Jersey (cited on pages 21, 22, 32, 40–42, 130).

[159] Andrew Trotman, Antti Puurula and Blake Burgess. 2014. Improvements to bm25 and language models examined. In *Proceedings of the 2014 Australasian Document Computing Symposium* (ADCS '14). ACM, Melbourne, VIC, Australia, 58:58–58:65 (cited on pages 6, 105).

[160] Christophe Van Gysel, Evangelos Kanoulas and Maarten de Rijke. 2017. Pyndri: a python interface to the indri search engine. In *Advances in Information Retrieval: 39th European Conference on IR Research, ECIR 2017*. Springer International Publishing (cited on page 113).

[161] Ellen M. Voorhees and Donna Harman, editors. *Proceedings of The REtrieval Conference (TREC 1-9)*, volume NIST Special Publication, (2001). National Institute of Standards and Technology (NIST) (cited on pages 105, 107).

[162] Ellen M Voorhees, Donna K Harman et al. 2005. *TREC: Experiment and evaluation in information retrieval*. Volume 63. MIT press Cambridge (cited on pages 2, 81).

[163] Nikos Voskarides, Dan Li, Andreas Panteli and Pengjie Ren. 2019. ILPS at TREC 2019 conversational assistant track. In *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019* (NIST Special Publication). Volume 1250. National Institute of Standards and Technology (NIST) (cited on page 10).

[164] Nikos Voskarides, Dan Li, Pengjie Ren, Evangelos Kanoulas and Maarten de Rijke. 2020. Query resolution for conversational search with limited supervision. In *The 43st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2020, Xi'an, China, July 25-30, 2020*. ACM (cited on pages 10, 93).

[165] Byron C Wallace, Issa J Dahabreh, Kelly H Moran, Carla E Brodley and Thomas A Trikalinos. 2013. Active literature discovery for scoping evidence reviews: how many needles are there. In *KDD workshop on data mining for healthcare (KDD-DMH)* (cited on pages 5, 32, 35).

[166] Lidan Wang, Minwei Feng, Bowen Zhou, Bing Xiang and Sridhar Mahadevan. 2015. Efficient hyper-parameter optimization for nlp applications. In *Proceedings of EMNLP*. Volume 15, 2112–2117 (cited on page 107).

[167] Ziyu Wang and Nando de Freitas. 2014. Theoretical analysis of bayesian optimisation with unknown gaussian process hyper-parameters. *arXiv preprint arXiv:1406.7758* (cited on page 110).

[168] William Webber, Mossaab Bagdouri, David D Lewis and Douglas W Oard. 2013. Sequential testing in classifier evaluation yields biased estimates of effectiveness. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 933–936 (cited on pages 35, 78).

[169] William Webber and Laurence A. F. Park. 2009. Score adjustment for correction of pooling bias. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '09). ACM, Boston, MA, USA, 444–451 (cited on pages 2, 4, 15, 16).

[170] William Edward Webber. 2010. *Measurement in information retrieval evaluation*. PhD thesis (cited on pages 1, 2).

[171] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan and Paul L Ruvolo. 2009. Whose vote should count more: optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, 2035–2043 (cited on pages 84, 92).

[172] Emine Yilmaz and Javed A. Aslam. 2006. Estimating average precision with incomplete and imperfect judgments. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management* (CIKM '06). ACM, Arlington, Virginia, USA, 102–111 (cited on pages 4, 16, 23, 24, 130).

[173] Emine Yilmaz and Javed A Aslam. 2006. Estimating average precision with incomplete and imperfect judgments. In *Proceedings of the 15th ACM international conference on Information and knowledge management*. ACM, 102–111 (cited on page 38).

[174] Emine Yilmaz, Evangelos Kanoulas and Javed A. Aslam. 2008. A simple and efficient sampling method for estimating ap and ndcg. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '08). ACM, Singapore, Singapore, 603–610 (cited on pages 4, 16, 23, 24).

[175] Emine Yilmaz, Evangelos Kanoulas and Javed A Aslam. 2008. A simple and efficient sampling method for estimating ap and ndcg. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 603–610 (cited on page 38).

[176] Dani Yogatama and Noah A Smith. 2015. Bayesian optimization of text representations. *arXiv preprint arXiv:1503.00693* (cited on pages 106, 107).

[177] Chengxiang Zhai and John Lafferty. 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, New Orleans, Louisiana, USA, 334–342 (cited on pages 6, 105).

[178] Chengxiang Zhai and John Lafferty. 2017. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Forum* number 2. Volume 51. ACM, 268–276 (cited on page 92).

[179] ChengXiang Zhai and John Lafferty. 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, Tampere, Finland, 49–56 (cited on page 107).

[180] Xueying Zhan, Yaowei Wang, Yanghui Rao and Qing Li. 2019. Learning from multi-annotator data: a noise-aware classification framework. *ACM Transactions on Information Systems (TOIS)*, 37, 2, 26 (cited on page 85).

[181] Haotian Zhang, Jimmy Lin, Gordon V Cormack and Mark D Smucker. 2016. Sampling strategies and active learning for volume estimation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 981–984 (cited on page 76).

[182] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan and Reynold Cheng. 2017. Truth inference in crowdsourcing: is the problem solved? *Proceedings of the VLDB Endowment*, 10, 5, 541–552 (cited on pages 82, 84).

[183] Yukun Zheng, Dan Li, Zhen Fan, Yiqun Liu, Min Zhang and Shaoping Ma. 2018. T-reader: a multi-task deep reading comprehension model with self-attention mechanism. *Journal of Chinese Information Processing*, 11, 128 (cited on page 11).

[184] Justin Zobel. 1998. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '98). Association for Computing Machinery, Melbourne, Australia, 307–314 (cited on page 2).

[185] Justin Zobel. 1998. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '98). ACM, Melbourne, Australia, 307–314 (cited on pages 2, 4, 15).

[186] Jie Zou, Dan Li and Evangelos Kanoulas. 2018. Technology assisted reviews: finding the last few relevant documents by asking yes/no questions to reviewers. In *The 41st International ACM*

SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018. ACM, 949–952 (cited on page 10).

Summary

The availability of test collections in Cranfield paradigm has significantly benefited the development of models, methods and tools in information retrieval. Such test collections typically consist of a set of topics, a document collection and a set of relevance assessments. Constructing these test collections requires effort of various perspectives such as topic selection, document selection, relevance assessment, and relevance label aggregation etc.

The work in the thesis provides a fundamental way of constructing and utilizing test collections in information retrieval in an effective, efficient and reliable manner. To that end, we have focused on four aspects around test collections for information retrieval.

We first study the document selection issue when building test collections (Chapter 2). We devise an active sampling method for efficient large-scale evaluation. Different from past sampling-based approaches, we account for the fact that some systems are of higher quality than others, and we design the sampling distribution to over-sample documents from these systems. At the same time, the estimated evaluation measures are unbiased, and assessments can be used to evaluate new, novel systems without introducing any systematic error.

Then a natural further step is determining when to stop the document selection and assessment procedure (Chapter 3). This is an important but understudied problem in the construction of test collections. We consider both the gain of identifying relevant documents and the cost of assessing documents as the optimization goals. We handle the problem under the continuous active learning framework by jointly training a ranking model to rank documents, and estimating the total number of relevant documents in the collection using a"greedy" sampling method.

The next stage of constructing a test collection is assessing relevance (Chapter 4). We study how to denoise relevance assessments by aggregating from multiple crowd annotation sources to obtain high-quality relevance assessments. This helps to boost the quality of relevance assessments acquired in a crowdsourcing manner. We assume a Gaussian process prior on the latent true labels to model the correlation between tasks. The proposed multi-annotator Gaussian process model is able to model the latent true labels, the task bias and variance, and the annotator bias and variance.

After a test collection is constructed, it can be used to either evaluate retrieval systems or train a ranking model. We propose to use it to optimize the configuration of retrieval systems (Chapter 5). We use Bayesian optimization approach to model the effect of a δ-step in the configuration space to the effectiveness of the retrieval system, by suggesting to use different similarity functions (covariance functions) for continuous and categorical values, and examine their ability to effectively and efficiently guide the search in the configuration space.

Samenvatting

De beschikbaarheid van testcollecties in het Cranfield-paradigma heeft de ontwikkeling van modellen, methoden en tools voor het vakgebied Information Retrieval aanzienlijk bevorderd. Dergelijke testcollecties bestaan doorgaans uit een verzameling van queries, een collectie van documenten en een bijbehorende relevantiebeoordelingen. Het samenstellen van deze testcollecties vereist werk vanuit verschillende perspectieven, waaronder het selecteren van queries, het selecteren van documenten, de relevantiebeoordeling en het samenstellen van relevantielabels, etc.

Het werk in dit proefschrift biedt een fundamentele aanpak om testcollecties op te bouwen en te gebruiken voor Information Retrieval op een effectieve, efficiënte en betrouwbare manier. Hiervoor hebben we ons gefocust op vier aspecten met betrekking tot testcollecties voor Information Retrieval.

Als eerste bestuderen we het probleem van documentselectie bij het opbouwen van testcollecties. We introduceren een active sampling-methode voor een efficiënte evaluatie op grote schaal. Anders dan bij andere active sampling gebaseerde methodes, houden we rekening met het feit dat sommige systemen van hogere kwaliteit zijn dan andere. Daarom hebben we een een sample-methode ontworpen die documenten uit kwalitatief goede systemen over-sampelt. Tegelijkertijd, zijn de nieuwe evaluation measures (op basis van een schatting) unbiased en kunnen die worden gebruikt om nieuwe, nieuwe systemen te evalueren zonder enige systematische fout te introduceren.

Een logische vervolg stap is om te bepalen wanneer de documentselectie en beoordelingsprocedure moet worden stopgezet. Dit is een belangrijk maar onderbelicht probleem bij het samenstellen van testcollecties. We beschouwen zowel de winst van het identificeren van relevante documenten als de kosten van het beoordelen van documenten als de optimalisatie-criteria. We benaderen dit probleem als een continue active learning-probleem, door gezamenlijk een ranking model te trainen om documenten te rangschikken en door het totale aantal relevante documenten in de collectie te schatten met behulp van een "greedy" sampling methode.

De volgende stap bij het samenstellen van een testcollectie is het beoordelen van de relevantie van de documenten. We bestuderen hoe relevantie-assessments kunnen worden verkregen die uit meerdere crowd-annotatiebronnen zijn samengevoegd. Dit om relevantie assessments van hoge kwaliteit te verkrijgen. Deze manier helpt om de kwaliteit van door crowdsourcing verkregen relevantie-assessments te verhogen. We gaan uit van een Gaussian-proces onderliggend aan de latent 'true labels' om de correlatie tussen taken te modelleren. Het voorgestelde Gaussian-procesmodel is in staat om de correct latente 'true labels' te modelleren, de taakbias en variantie en de annotatorbias en variantie.

Nadat een testcollectie is samengesteld, kan deze worden gebruikt om retrieval systems te evalueren of ranking modellen te trainen. We stellen voor om deze documentencollectie te gebruiken om de configuratie van retrieval-systemen te optimaliseren. We gebruiken de Bayesiaanse optimalisatietechniek voor het modelleren van het effect van een δ-step in de configuratie-space op de effectiviteit van het retrievalsysteem. Dit door gebruik te maken van verschillende gelijkenisfuncties (covariantiefuncties) voor continue en categorische waarden, en hun vermogen om effectief en efficient de optimalisatie van de configuratie te sturen.