EXPLORING INFORMATION RETRIEVAL APPROACHES FOR CLINICAL DECISION SUPPORT AND BIOMEDICAL SEARCH TASKS

by

Mohammad Reshood Alsulmi

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Summer 2018

EXPLORING INFORMATION RETRIEVAL APPROACHES FOR CLINICAL DECISION SUPPORT AND BIOMEDICAL SEARCH TASKS

by

Mohammad Reshood Alsulmi

Approved: ____
Kathleen McCoy, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: ____
Babatunde Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved: ____
Douglas J. Doren, Ph.D.
Interim Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ____

Benjamin Carterette, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ____

Cathy Wu, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ____

Hui Fang, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ____

James Clause, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

First of all, I would like to express my most profound gratitude towards my advisor and mentor Prof. Ben Carterette who introduced me to the field of information retrieval and guided me throughout my Ph.D. journey. Prof. Ben taught me the correct way to conduct scientific experiments to test and validate my ideas and gave me the freedom to explore diverse research directions. I am extremely thankful for him for dedicating considerable time and energy to discuss my ideas and to provide me with a countless number of comments and suggestions. It would not be possible to reach this point without his support and help.

Secondly, I would like to thank Prof. Cathy Wu, Prof. Hui Fang, and Prof. James Clause for serving my dissertation committee, sharing their time and expertise, and providing me with insightful comments and suggestions. Also, I am grateful to my friends and colleagues at the IR lab for their advice and generous support. The good times and memories we enjoyed together while attending conferences and sharing ideas will never fade.

I would like to thank King Saud University for the generous financial support for my family and me throughout my Ph.D.

Special thanks to my dear wife Norah Alsogaih who did an excellent job taking care of my two little daughters and me during my studies. Also, I want to thank my parents, my mother Modhi Alfuhaid and my father Reshood Alsulmi, for their love, encouragement, support, and understanding.

TABLE OF CONTENTS

Chapter

Appendix

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Health care practitioners seek out relevant information about providing the best possible care for their patients. Relevant information to those practitioners can vary between finding potential diagnoses, suggesting related lab tests, and determining treatments for a given patient. The role of a clinical search system, relying on information retrieval (IR), is to exploit the enormous volume of publications in the biomedical literature and make them efficiently accessible to those practitioners. In this thesis, we propose to develop several approaches which can improve the effectiveness of search results produced by clinical search systems. Our approaches can help in minimizing the overhead of manual settings in current search approaches and reduce the complexity of search tasks performed by physicians.

We propose to design a clinical search system that extends traditional search strategies by incorporating relevant biomedical domain resources. In our system, we will rely on some essential retrieval models (e.g., Divergence From Randomness, Pseudo-Relevance Feedback) and some text reformulation tasks that are based on domain-specific knowledge (e.g., Unified Medical Language System, Medical Subject Headings) in order to improve our search strategy. Moreover, we show that our system can be further enhanced when using simulated user models in evaluating the relevance of medical concepts appearing in clinical search queries.

Lastly, we study the usefulness of adopting learning to rank (LtR) framework for clinical search tasks. We propose a methodology for evaluating several LtR models which includes relying on a diverse set of training features as well as some feature selection techniques and show how this method can lead to learning effective ranking models for clinical search.

Chapter 1

INTRODUCTION

Information Retrieval (IR) has the benefit of being widely applicable to many domains and information seeking tasks. A variety of IR techniques have been successfully adopted for a wide range of search tasks with the goal of locating relevant information resources (e.g., web pages, images, videos, audio clips, etc.) that satisfy certain search queries provided by users. In particular, for general and professional health care, there has long been an interest in using IR in the search tasks for these domains. In those settings, health care providers such as physicians perform search tasks to seek out relevant information about providing the best possible care for their patients.

Relevant information resulting from the seeking tasks performed by clinicians can vary between finding potential diagnoses, suggesting related physical and lab tests, and determining effective treatments for a given patient [104]. The role of clinical search systems relying on IR methods is to exploit the enormous volume of publications in biomedical literature by making them efficiently and effectively accessible to health care providers. However, several challenges are present when performing search tasks in clinical settings.

One challenge is the complexity of search tasks performed by clinicians. Unlike general web search users who write short queries with two or three words [57], physicians tend to write relatively long queries—summaries of medical cases—with detailed information about their patients. The type of information expected in those cases includes some demographic attributes (age and gender), main clinical findings (e.g., symptoms, diseases, genetic profiles, lab and physical tests, etc.), and any other known medical history. The length of clinical queries can present a challenge to search systems in that there is a little or no information provided to these systems on how to effectively interpret those cases or which part of those cases to focus on in order to sufficiently meet user needs.

Another challenge is the ambiguity of clinical concepts and terms that are present in medical cases. In particular, the medical language used to indicate some terminologies and conditions is very different from what is used in general English. For instance, terminologies such as “hypertension”, “HBP”, and “high blood pressure” are known by domain specialists to refer to the same clinical concept; however, such knowledge may not be observed by traditional search systems [70]. In fact, this limitation in handling ambiguous medical terms is expected to have a direct impact on the quality of search results (particularly, precision and recall) which eventually can lead to retrieving less relevant information to the users.

Lastly, the diversity of the clinical and biomedical search tasks also presents a challenge as different tasks usually need specific design and implementation of their search methods. We would expect that no single approach can fit all of the different search tasks, but the generalization of some approaches like the combination of IR techniques with some other learning techniques and applying some query modification approaches can be helpful in accomplishing those different tasks.

Looking at prior work, we see that there are several approaches which aim to tackle some of the challenges that we discussed above [120, 88, 56, 135, 70, 122]. We will discuss a variety of these approaches along with some related work in Chapter 2. Similar to these approaches, in this thesis, we focus on addressing those challenges and limitations by conducting several studies with the goal of improving search results and reducing the complexity of clinical search tasks. Generally, we focus on studying the aspects which can have a significant impact in increasing the effectiveness of search results (i.e., the quality of results returned to users) in addition to some aspects concerning the efficiency of the systems used to produce these results. Our studies are briefly explained in the following four sections.

1.1 Query Reformulations for Clinical Search Tasks

Because medical cases, which are used to inquiry search systems, are complex and usually contain some ambiguous words and phrases, reformulation techniques such as removing less important words, boosting other important words, and adding some related names and synonyms are expected to be a key player in enhancing search results [1, 2, 70, 71, 120]. In the study described in Chapter 3, we evaluate two kinds of these techniques: query refinements, by applying a set of modification such re-weighting and eliminating some query terms, and ontology-based expansion, performed by adding related terms to user queries from medical and gene ontology.

We explore these techniques by applying them to two different types of clinical search tasks: tasks for clinical decision support, and search tasks for precision medicine. In our experiments, we show that applying those techniques and specifically those for refining clinical queries (by removing and re-weighting a selected set of query concepts) can lead to significant improvement over baseline clinical search system. Moreover, expanding key terms in those queries with other concept names using a controlled vocabulary in many cases is shown to reduce the complexity of a given search task and improve search results.

1.2 Framework for Automatic Query Reformulations

One main limitation in the previous methods for query reformulations is that they heavily rely on a manual effort in settings some of the techniques and strategies used in these approaches. For instance, in the methods proposed by Limsopatham et al. [71] and Palloti et al. [88], the authors needed to manually select distinct sets of semantic types to filter the words and phrases which appear in clinical queries. Aiming to tackle this limitation, we present a framework for transforming this manual process into a fully automatic one using the advancement of Machine Learning (ML) techniques. Our framework relies on simulated physician feedback to learn models that can predict if a particular concept (i.e., term or phrase) in a query is clinically relevant or not. In Chapter 4, we discuss this framework and then demonstrate how to use it to predict concept importance rating with high accuracy.

1.3 Learning to Rank for Clinical Search

In contrast to traditional information retrieval models which combine few primitive features such as term and inverse document frequencies, learning to rank (LtR) models can combine a large number of features and use them to perform ranking [72]. While several studies have shown the success of the adaptation of LtR in general web search [76, 91, 14], in our study described in Chapter 5, we focus on applying these models for clinical search tasks and examining their effectiveness for such domain-specific tasks.

We rely on a diverse set of features that exemplify both general and domain-specific statistics and use them to learn several models for document ranking. Moreover, since the number of training features could increase quite a lot which might be leading to learning ineffective models, we evaluate several feature selection strategies and show that LtR learning performance can be optimized when considering greedy feature selection. In overall, our study suggests that learning to rank models can promote search results in clinical settings with high effectiveness compared to traditional retrieval models.

Chapter 2

RELATED WORK

In this chapter, we briefly describe some related work in the field of information retrieval. First, in Section 2.1, we provide a brief background in information retrieval (IR) and some related topics such as ranking models and IR evaluations. Then, in Section 2.2, we move to provide an overview of prior work applying IR in the context of the clinical and biomedical search.

2.1 Information Retrieval

Information Retrieval (IR) can be defined as the process of finding relevant material (usually in the form of documents) that satisfies a given information need (in the form of queries) from a large collection of information resources (e.g., web pages, scientific articles, etc.) [77]. In practice, commercial web search systems can be seen as good examples of IR applications that accept user queries and generate responses representing a wide range of information resources (e.g., web pages, images, videos) [36, 63]. Our work, in this thesis, relies on retrieval models for text-based documents to obtain relevant clinical data, so we start by reviewing some of these models.

2.1.1 Basic Retrieval Models

We consider five fundamental retrieval models: Boolean model, vector space model, BM25 model, language model, and divergence from randomness model. These models are reviewed below.

2.1.1.1 Boolean Retrieval Model

Boolean model is one basic retrieval model in which query terms are combined by the Boolean operators AND, OR, and NOT [66, 77]. The model views each document as a set of words and returns an unordered list of documents as a response to each given query. The returned documents from the search with this model are expected to satisfy the Boolean operators defined in the given query.

2.1.1.2 Vector Space Retrieval Model

Vector space model allows free text queries (i.e., without Boolean operators) and returns a list of ordered documents as a response to a given query [100]. In this model, both documents and queries are represented by the vectors of their term weights. TF-IDF is a common way to estimate vector weights in which TF represents the number of occurrences of a term within a document whereas IDF is the inverse document frequency [107] which is computed as:

$$IDF_t = \log \frac{N}{df_t} \tag{2.1}$$

where N is the collection size (the number of documents), and df_t is the document frequency of a term t which represents the number of documents in the collection which contain the term t. The vector space model ranks the documents in the collection by computing a relevance score for each document. The relevance score usually is computed using cosine similarity between query and document vectors.

2.1.1.3 BM25 Model

Okapi BM25 is another basic retrieval model that is commonly used in IR [61]. It relies on a probabilistic ranking function that is defined as:

$$Score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{tf_{q_i,D} \cdot (k_1 + 1)}{tf_{q_i,D} + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \tag{2.2}$$

where Q is a query with n terms, $|D|$ is the document length in terms, avgdl is the average document length in the whole collection, and k_1 and b are free parameters. The computation of IDF is slightly different from that of TF-IDF and defined as:

$$IDF(q_i) = \log \frac{N - df(q_i) + 0.5}{df(q_i) + 0.5} \tag{2.3}$$

where N is the collection size, and $df(q_i)$ is the the number of documents in the collection which contain the term q_i.

2.1.1.4 Language Model

Language model (LM) is a probability distribution over streams of words and other linguistic units in a language [90, 55, 15, 81]. Query likelihood, which is a very basic model, is part of language model and most commonly used for information retrieval. In this model, document score is estimated by computing the probability of the document being relevant given a query. Smoothing techniques are usually used in LM model to estimate the probability of unseen words in the documents [25, 130]. The scoring function of query likelihood language model is computed as:

$$Score(D,Q) = \log P(Q|D) = \sum_{i=1}^{n} \log \frac{tf_{q_i,D} + \mu \frac{tf_{q_i,D}}{|C|}}{|D| + \mu} \tag{2.4}$$

where Q is a query with n terms, $|D|$ and $|C|$ are the document and collection lengths in terms respectively, $tf_{q_i,D}$ and $tf_{q_i,C}$ are the document and collection frequencies of query term q_i respectively, and μ is the Dirichlet smoothing parameter [90, 130].

2.1.1.5 DFR Model

Divergence from Randomness (DFR) is a class of non-parametric probabilistic retrieval models [9, 8]. In these models, terms weights are estimated by measuring the divergence between an actual term distribution and a distribution produced by a random process. The idea behind such models is that as more the divergence of a term frequency within a document from its frequency within the collection frequency, as more informative that term in the documen t[9, 8].

DFR models computations are carried by performing the three steps: choosing the basic randomness model, computing the first normalization, and lastly, applying the normalization of term frequencies [8]. For instance, computing document scoring function with inverse expected document frequency randomness ($InExpB2$) basic model can be defined as:

$$Score(D,Q) = \sum_{t \in Q} qtw \cdot w(t,D) \tag{2.5}$$

where t is a term in query Q, qtw is the term weight in Q, and $w(t, d)$ is the term weight in document d estimated as:

$$w(t, D) = \frac{F + 1}{n_t \cdot (tfn + 1)} \Big(tfn \cdot \log_2 \frac{N + 1}{n_e + 0.5} \Big) \tag{2.6}$$

tfn and n_e in the equation above are estimated as described in [8].

2.1.2 Advanced Retrieval Models

Apart from the basic models, several advanced techniques and models have been proposed in IR. In this section, we focus on describing some of these techniques which we consider in our work such as Learning to Rank (LtR) and Pseudo-Relevance Feedback (PRF).

2.1.2.1 Learning to Rank

Learning to Rank (LtR) can be defined as all methods that use machine learning (ML) to estimate document ranking [72, 68]. Compared to traditional IR models which combine few primitive features, learning to rank algorithms can learn the optimal way for combining a large number of features extracted from query-document pairs using discriminative training. Those algorithms can be categorized according to input space, output space, loss function, and hypothesis space into three approaches: the pointwise approach, the pairwise approach, and the listwise approach.

Pointwise algorithms are trained to predict the ground truth label (i.e., the relevance) of each document regardless of the inter-dependency between documents. Examples of pointwise algorithms include random forests [16], Pranking [35], and McRank [69]. On the other hand, pairwise and listwise learners take such inter-dependency into consideration such that in pairwise learners (e.g., RankBoost [41], LambdaMart [126], and RankNet [21]), the ground truth labels are predicted for each pair of documents (i.e., the relative order between two documents) whereas in listwise algorithms such as AdaRank [127], ListNet [23], and Coordinate Ascent [80], a list of retrieved documents is chosen from a document permutation such that those documents are predicted to be relevant.

2.1.2.2 Pseudo Relevance Feedback

Relevance feedback refers to the process of involving the user in the retrieval task with the goal of improving the final result list [77]. More specifically, the user is expected to provide feedback on which documents are relevant using initial results set, and then the search system uses the provided feedback to revise the final result list. Although relevance feedback models may not be considered as retrieval models, combining these models with basic retrieval models such as BM25, LM, and DFR is expected to leverage the performance of search systems [98, 99, 18].

Pseudo-Relevance Feedback (PRF), on the other hand, is different from direct relevance feedback in that it is purely blind and does not involve any user feedback. Instead, the retrieval system assumes that the initial result set contains relevant documents (i.e., the top k documents), so it selects the terms with the highest weights from these documents and uses them to refine and expand the original user query. Examples of PRF approaches include Rocchio's PRF model [18], Lavrenko's relevance model [67], and KL divergence [9].

2.1.3 Evaluations in IR

Evaluating retrieval effectiveness, by studying the quality of search results, plays a crucial role in IR as it provides researchers with useful tools for exploring different aspects of search system's design, development, and maintenance which ultimately can lead to building effective retrieval systems [30]. A strong focus of IR evaluations has been on measuring system effectiveness (i.e., validating whether a retrieval strategy or algorithm can discriminate between relevant and non-relevant documents for a given user query) [101]. This can be accomplished by performing evaluations using test collections, comprised of a set of documents, a set of queries, and a set of relevance information for documents to given queries [137, 116]. In the section below, we provide an overview of these test collections, and later, we review and describe some evaluation measures that are widely adopted in IR.

2.1.3.1 Test Collections

Test collections provide an ideal environment to perform IR experiments as they give the researchers a way to evaluate different retrieval strategies and to study key factors that contribute to retrieval effectiveness of search systems[101]. Generally, those collections are generated as a part of large user study experiments which involve creating test topics, generating sample documents, collecting search results from participants, and generating relevance assessments [116, 137]. Over the years, several test collections have been developed to motivate researchers in addressing search challenges for a variety of domains ranging from very broad (e.g., general web search [28, 39] , tasks and suggestions [24, 115]) and to very specific problems (e.g., legal search [53], cross-language search [43], medical and healthcare [118, 104]).

2.1.3.2 Evaluation Metrics

In this section, we describe some of the commonly used evaluation metrics in IR. Mainly, we consider six evaluation metrics: precision, recall, R-prec, MAP, NDCG, and bpref.

Precision and Recall

Precision (P) is a basic measurement for evaluating the effectiveness of the results obtained by a retrieval system, and it represents the fraction of relevant documents retrieved from the total number of documents [92]. Precision is a set-based measure which means it is computed for unordered sets of documents, but due to the fact that evaluations in IR are typically based on ordered lists of documents, we compute this measurement at a specific ranking position k (e.g., 5 or 10) [30, 77]. As a result, we can define precision as the proportion of relevant documents among the top k documents retrieved.

$$P@K = \frac{\#(\text{relevant documents retrieved at K})}{\#(\text{documents retrieved at K})} \tag{2.7}$$

Recall (R) is another basic measurement used for evaluating the results produced by search systems. It represents the proportion of retrieved relevant documents to the

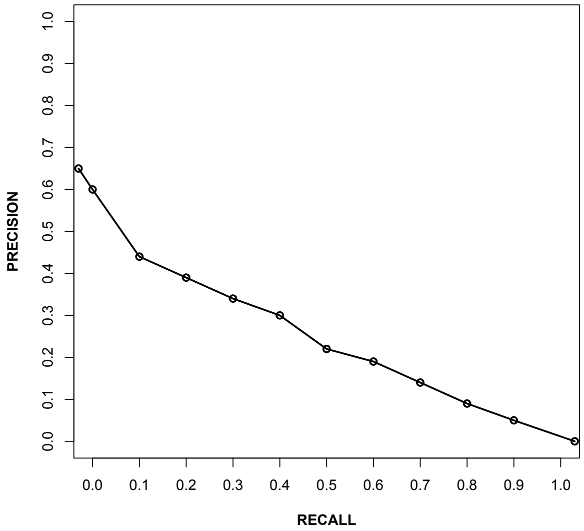


Figure 2.1: Sample precision-recall graph.

total number of the relevant documents in the whole test collection, and similar to precision, it can be computed at each position of the ranked results [77, 92].

$$R@K = \frac{\#(\text{relevant documents retrieved at K})}{\#(\text{all relevant documents})} \tag{2.8}$$

To study the relationship between precision and recall for a certain IR system or to compare several systems, we can plot the two measurements against each other using the precision-recall graph created using 11 cutoff values from recall level precision averages. These graphs are expected to be falling downward from left to right which confirms that the two measures are inversely related, meaning that as more relevant documents are retrieved, the more non-relevant documents are retrieved [84]. Figure 2.1 shows a precision-recall graph plotted to evaluate a sample IR system.

R-Precision

Although measuring precision at fixed rank position (e.g., P@10) correlates well with user satisfaction in search tasks, in some cases, it may lead to poor discrimination of different retrieval methods [19, 20]. Thus, an alternative way to compute precision is R-prec which measures the precision after R_q documents are retrieved, where R_q is the total number of relevant documents for query q [30].

MAP

Mean Average Precision (MAP) is one of the commonly used measures in IR which provides a single-figure measure of quality across recall levels [77, 20]. MAP can be obtained by computing the mean of precision values for each recall level (i.e., after each relevant is retrieved) as shown in the following formula.

$$MAP(Q) = \frac{1}{|Q|} \sum_{q \in Q} \frac{\sum_{q \in \{d_1 ..., d_j\}} Precision(rank(d))}{|\{d_1 ..., d_j\}|} \tag{2.9}$$

where $\{d_1 ..., d_j\}$ is the set of relevant documents for query $q \in Q$ such that Q is the set of all queries in a given test collection.

NDCG

Normalized Discontinued Cumulative Gain (NDCG) is a measure of ranking quality which considers the situations where a graded relevance scale of documents is used (e.g., relevant=2; partially-relevant=1; non-relevant=0) [77, 58]. The measure assumes that users would prefer to see highly relevant documents appearing in the top of ranking lists. Also, it assumes that relevant documents with higher relevance degree are far more useful than marginally relevant documents, which in turn are more useful than non-relevant documents [92]. Similar to $P@K$, NDCG can be measured at specific ranking position k (usually 10, or 20). For a given set of queries Q, NDCG can be computed as:

$$NDCG(Q, k) = \frac{1}{|Q|} . \sum_{q \in Q} \frac{1}{IDCG} . \sum_{i=1}^{k} \frac{2^{R(q,d_i)} - 1}{log(1 + i)} \tag{2.10}$$

where q is a query in the set Q, $IDCG$ is the ideal discounted cumulative gain, $R(q, d_i)$ is the relevance degree of document at rank i, and $log(1 + i)$ is the discounting factor.

bpref

bpref, which stands for binary preference, is a measure of retrieval effectiveness which is widely used in the scenarios where the relevance judgments are known to be incomplete or when there are few judged documents [19, 29, 128]. bpref is based on the relative ranks of judged documents only such that it defines a preference relation which prefers judged relevant documents to be retrieved before judged not relevant documents [84]. The bpref measure is defined as follows:

$$\text{bpref}(q) = \frac{1}{R}.\sum_{r}(1 - \frac{|n \text{ ranked higher than } r|}{min(R, N)}) \tag{2.11}$$

where R is the number of judged relevant documents, N is the number of judged non-relevant documents, r is a retrieved relevant document, and n is the number of R retrieved documents that are not relevant.

2.2 Clinical Based Retrieval

In this section, we discuss and highlight some prior work for applying IR methods in a variety of clinical and biomedical search tasks. We first present and briefly describe several tracks for simulating search tasks performed by clinicians in finding relevant clinical data for providing better healthcare to their patients (essentially, we consider the tasks and data sets offered by TREC). Then, in the second part of this section, we discuss some previous work which focuses on examining different IR search strategies and approaches to increase the effectiveness of search results in the scope of the clinical domain.

2.2.1 Simulating Clinical and Biomedical Search Tasks

The Text Retrieval Conference (TREC) is an annual meeting that is organized by the National Institute of Standard and Technology (NIST) with the aim of providing a platform for performing experiments and evaluations of search systems which eventually lead to improving the state-of-the-art IR methodologies. Each year, the organizers of the conference introduce a set of tracks (i.e., competitions) to challenge participants, from academia and industry, with specific research problems. Over the years, several challenges have been proposed to motivate the exploration of IR methods for search and information seeking tasks in the biomedical domain. In this part, we briefly describe some of those tracks that simulate search tasks performed in clinical settings.

2.2.1.1 TREC Medical Records Track

The goal of the Medical Records track is to encourage the research community to explore a variety of methods for effective electronic health records (EHRs) retrieval [119, 118]. The task provided by the track's organizers was an ad hoc retrieval task which attempts to simulate search scenarios which clinical researchers perform to identify cohorts for comparative effectiveness research or for designing clinical studies/trials. In those scenarios, clinical researchers usually need to design inclusion criteria which define the group of patients who are eligible and will be included in that study. The inclusion criteria are expected to contain some specific attributes such as age, gender, and ethnicity of the participating patients as well as some essential information about the medical condition of those patients (e.g., diseases and treatments) [117].

The organizers of the track simulated the described task by providing the participating groups with a sample of clinical topics representing a variety of inclusion criteria for clinical studies (Table 2.1 shows several examples of these topics) and a set of de-identified medical records provided through the University of Pittsburgh NLP Repository (originally, they were collected from several hospitals). The participants in the track were expected to explore several retrieval strategies and then submit their results, consisting of lists of visits ranked by the likelihood that a patient's visit has satisfied given inclusion criteria, to the organizers in order to be judged (by human assessors) and evaluated. Generally, a winning retrieval strategy is expected to find more relevant patient's visits and thus more effective search results. Unfortunately, due to some privacy concerns, the track was discontinued after two years of participation.

ID	Topic
109	Women with osteopenia.
136	Children with dental caries.
160	Adults under age 60 under going alcohol withdrawal.
169	Elderly patients with subdural hematoma.

Table 2.1: Example topics from the TREC Medical Records Track test set.

2.2.1.2 TREC Clinical Decision Support Track

TREC launched the Clinical Decision Support (CDS) track in order to simulate search tasks performed by physicians while they are providing health care to their patients (e.g., search for relevant biomedical data which can support in diagnosing or treating a given patient) [104, 96, 93]. The goal of the track is to encourage the application of different algorithmic approaches to the problem by participating research groups (from both academia and industry) spanning the world in order to understand which methods are most effective. One fundamental difference between this track and the Medical Records track described in Section 2.2.1.1, is that the task introduced in this track focuses on performing evidence-based clinical decision whereas in the Medical Records track it focuses on gathering profiles about patients that are candidates for a particular clinical study (consequently, biomedical articles will be more suitable for clinical decision support than medical records).

The organizers of this track provided the participants with a collection, featuring a large sample of biomedical articles, and real clinical cases, which represent typical information seeking tasks requested by clinicians. Each case represents a summary of a patient's medical history including findings, lab test, and if any previous treatments. Table 2.2 illustrates a sample of these topics provided by this track.

The track lasted for three years from 2014–2016 where for each year, about 30 participating research groups used the provided collection and several IR techniques to build their systems in order to answer a number of test queries (30 queries per year). The results from each participating group comprise lists of the top retrieved articles for each case, which are submitted to the track organizers and then assessed by human assessors (domain experts) to show which of those resulted articles are relevant and which are not. Moreover, to test more system features, the organizers allowed each participant to submit multiple trials which resulted in more than 90 submissions for each year. Once the various approaches have been evaluated, a more in-depth analysis can reveal which techniques tend to work in general and which do not [95].

Topic Type	**Clinical Summary**
Diagnosis	58-year-old woman with hypertension and obesity presents with exercise-related episodic chest pain radiating to the back.
Test	40-year-old woman with severe right arm pain and hypotension. She has no history of trauma and right arm exam reveals no significant findings.
Treatment	63-year-old heavy smoker with productive cough, shortness of breath, tachypnea, and oxygen requirement. Chest x-ray shows hyperinflation with no consolidation.

Table 2.2: Sample topics from the TREC CDS Track shown as summarized clinical cases.

2.2.1.3 TREC Precision Medicine Track

The TREC Precision Medicine (PM) track (introduced in 2017) is motivated by the research problem of developing effective search systems for finding treatment plans based on an individual's genetic, environmental, and lifestyle profile [94]. Similar to the Clinical Decision Support (CDS) track, the Precision Medicine (PM) track focuses on assisting clinicians (mainly oncologists) to retrieve relevant biomedical articles which help them in making clinical decisions in regards to their patients. However, it differs from the CDS track in that the proposed task is more specific to finding scientific articles which assist in finding suitable treatments for patients with cancer diseases (i.e., PM track extends the previous CDS track with more focus on specific clinical questions compared to the general clinical decision support) [4].

The topic set for this track were provided in the form of structured clinical cases such that each case will define precise patient's profile which consists of demographic attributes (e.g., age and gender), cancer type, gene variants, and potentially other information about the patient (e.g., other diseases and health concerns). Table 2.3 shows several examples of the topics that are used on the PM track.

The participants in this task were challenged to retrieve relevant biomedical abstracts which contain scientific evidence that can be relevant in treating a given patient. Additionally, a secondary task was included in which the participants were asked to retrieve the top clinical trials which a given patient might be eligible for. Like other TREC challenges, this track applies similar evaluation procedures which involve relying on human assessors to judge the results submitted by participating groups.

ID	Patient Case
1	*Disease:* Liposarcoma *Gene:* CDK4 Amplification *Demographic:* 38-year-old male *Other:* GERD
4	*Disease:* Breast cancer *Gene:* FGFR1 Amplification, PTEN (Q171) *Demographic:* 67-year-old female *Other:* Depression, Hypertension, Heart Disease
12	*Disease:* Colon cancer *Gene:* BRAF (V600E) *Demographic:* 35-year-old male *Other:* None

Table 2.3: Example topics from the TREC Precision Medicine Track shown as structured clinical cases.

2.2.2 Query Reformulations

Prior work on query reformulations focuses on modifying clinical queries by either applying query refinements or performing ontology-based expansion. In the following section, we discuss the two kinds of previous research work.

Query Refinements

Semantic mapping has been used to improve query representation by applying a set of refinement techniques. For instance, Limsopatham et al. [71] introduce query refinement approach which combines bag-of-words (BoW) and bag-of-concepts (BoC) representations (i.e., each query is represented using both vector of words and vector of numerical identifiers). In this approach, semantic mapping is used to identify clinical concepts that are relevant to the task of the query and remove others that do not match the query task. Likewise, Palotti et al. [88] apply a refinement approach that is based on re-weighting clinical concepts. It considers only a specific clinical concepts for re-weighting. (i.e., concepts that belong to 34 UMLS semantic types out of 133 types). Compared to Limsopatham et al., this approach uses BoW only. Also, another work such as that of Wan et al. [120] and Hu et al. [56] performs manual query refinements in a way that physicians are asked to refine a set of clinical queries (i.e., re-write queries by adding and removing some terms). The results from this process show a major performance improvement, suggesting that query refinements indeed improve the retrieval; however, one drawback is that this approach is performed manually.

Our query refinement approach in this work is distinguished from previous work in that we consider several refinement tasks besides removing less important terms. In particular, we investigate field-based score boosting and concept re-weighting. In addition, we attempt to automate applying these tasks by introducing a semantic-based approach which relies on a learned model to select the appropriate refinement actions.

Ontology Based Expansion

Another type of work uses ontology-based expansion to reduce the ambiguity of concepts in clinical queries. Zhu et al. [135] use MeSH to expand each clinical concept with its entry terms and descendant concepts in MeSH tree. Then, a user query log is used for weighting the expanded concepts before submitting the query to a CDS system. Another work by Mourão et al. [82] also uses MeSH for expansion; however, instead of using user logs to weight the expanded concepts, a sample of 350,000 biomedical articles is used for estimating weights. One limitation of both approaches is that they do not properly map some conceptual dependencies. In particular, Zhu et al. consider all descendant concepts in MeSH tree to be eligible for expansion which may cause query drifting especially when the expanded concepts are quite broad (i.e., in the top of the tree). Also, both Zhu et al. and Mourão et al. do not map the conceptual dependencies between sibling concepts in MeSH tree and eliminate them from expansion.

Aiming to overcome this limitation, Limsopatham et al. [70] propose an approach which applies text mining techniques on several biomedical resources (including MeSH) to estimate the relevance of each concept that is a candidate for expansion. Our work in this thesis aims to address the same limitation; however, instead of using external resources, we apply domain specific similarity measurement that uses MeSH tree structure to select eligible concepts for expansion. Our approach is shown to improve the performance with high retrieval effectiveness.

2.2.3 The Effectiveness of Learning to Rank

Much prior work on LtR for IR focuses on either proposing new supervised machine learning algorithms that are suitable for LtR (e.g., LambdaMart by Wu et al. [126] and Coordinate Ascent by Metzler et al. [80]) or adapting some of the previously proposed algorithms into some practical search scenarios by introducing new sets of features and evaluating their effectiveness for various search tasks [91, 75]. We observe that for some domains there is still very little work considering applying LtR in their search tasks. In particular, for clinical and general health purposes, we are only aware of the two approaches by Palotti et al. [87] and Soldaini et al. [106] that apply LtR in consumer health search while examining a variety of features aiming to a better understanding of user queries. Both approaches focus on promoting LtR for the tasks where lay people (not medical experts) use web search engines to look for health content such as health advice (i.e., not for providing professional healthcare).

Our work, on the other hand, aims to examine the application of LtR to the search tasks performed by professional health care providers such as physicians. Therefore, we use the publicly available data set by the TREC Clinical Decision Support (CDS) track [93], which contains a sample of medical cases and a collection of biomedical articles, in order to simulate those search tasks. In addition, because LtR model learning can be affected by the kind of features used in building such models, we train with a comprehensive set of features that exemplify a variety of statistics used in different IR tasks (such as LETOR [91] and query performance predictors [52]) as well as some specific features for biomedical search [71]. These features aim to capture different aspects about the cases and the articles in our dataset that are not captured by traditional IR features.

Also, because learning with a large number of features could lead to overfitting, especially when the training data contains a small number of samples, we examine the use of Feature Selection (FS) in optimizing model training and enhancing the generalization of the learned models. FS helps in learning highly effective ranking models with low computational cost when considered for large-scale learning scenarios, whereas in small-scale scenarios, FS can reduce the overhead of overfitting which leads to more generalization of the learned models. FS techniques are classified into *filter*, *wrapper*, and *embedded* methods [47]. A *filter* method is performed at pre-training stage by computing an importance score for each feature and selecting those features which have the highest importance among all features. *Wrapper* methods search a space of feature subsets and then select a feature subset that maximizes the performance of a learned model. Lastly, *embedded* methods perform feature selection as an integrated part of the learning process.

Due to the importance of FS, some studies have been conducted to evaluate its effectiveness within the scope of LtR. Geng et al. [42] propose a filter based Greedy Search Algorithm (GAS) that performs an iterative procedure to select feature subsets that maximize relevance and minimize similarity among all features. Dang et al. [37] propose a selection method that is based on coordinate ascent and best-first-search in which greedy search is used to combine non-overlapping feature subsets and reduce them to fewer feature subsets to be used for learning. Recently, Gigli et al. [44] propose several feature selection methods such as NGAS and XGAS which rely on greedy search to select feature subsets with the highest importance and minimum similarity.

Our preliminary experiments with some of the methods discussed above show a poor generalization of these techniques when considered for medical search tasks. This can be addressed to the fact that our training set is relatively small (small-scale training) whereas most of these methods have been designed to target large-scale web learning tasks (in other words, the majority of these methods focus on optimizing learning time rather than reducing the effect of overfitting). Aiming to tackle this limitation, we examine a wrapper FS method which extends the GAS algorithm for greedy search. Our method is shown to work effectively for small-scale data settings and significantly outperform some of the state-of-the-art filter methods.

Chapter 3

CLINICAL CASE SEARCH WITH QUERY REFORMULATIONS

As described in Chapter 1, one of the challenges when performing search tasks in clinical settings is that clinical practitioners usually issue long queries with many words describing detailed information about patients. Those long queries may present some complexity to search systems which can lead to retrieving less relevant articles. Another challenge is the ambiguity of clinical terms used by clinicians. For instance, terminologies such as "hypertension" and "high blood pressure" are known to refer to the same medical concept; however, such knowledge is not observed by traditional retrieval systems [70].

In this chapter, we aim to tackle some of these limitations by examining two query reformulation approaches [2]. First, we apply query refinements by removing less important terms and performing query re-weighting. We assume that some query terms may not be clinically related, so they either need to be removed or have their weights reduced, whereas other terms need to be boosted due to their importance in answering the clinical query. Second, we rely on ontology-based term expansion using Medical Subject Heading (MeSH)[1] to reduce the ambiguity of clinical terms in clinical queries. Moreover, we develop a simple approach which relies on semantic mapping in order to recommend suitable actions to be performed to each query term (i.e., whether to remove, re-weight, or expand that term). Our approach is performed automatically (i.e., no human interaction is needed after an initial assignment of semantic types to bags). In our evaluation, we show that our proposed approaches can achieve over 40% improvement compared to a strong baseline.

[1] https://meshb.nlm.nih.gov/search

3.1 Organization of this Chapter

In Section 2.2.1, we presented several examples of search and information seeking tasks that are typically performed by medical researchers and practitioners. In this chapter, we focus on exploring a variety of IR approaches by considering two types of clinical search tasks: clinical decision search tasks, and search tasks for precision medicine. The first set of tasks is provided as unstructured clinical summaries and focuses on answering a wide range of clinical questions regarding providing healthcare to patients (i.e., not specific to a certain medical condition or disease). On the other hand, the tasks for precision medicine are more specific to finding potential treatments for cancer patients performed based on a patient's genetic profile. Thus, they tend to be more precise and contain some structured information. Table 3.1 summarizes the key differences between clinical decision support and precision medicine tasks that are simulated using TREC settings. The organization of this chapter proceeds as follows. In Section 3.2, we provide an overview of clinical decision search tasks and describe the data sets we use to simulate those tasks. Furthermore, we propose and examine several reformulation approaches and show how to improve the quality of results by performing those tasks. Later, in Section 3.3, we introduce search tasks for precision medicine, and then we discuss how to promote search results for these tasks by extending some of the proposed approaches for clinical decision support.

	Clinical Decision Search Tasks	**Precision Medicine Tasks**
Domain	General practice	Specific to cancer patients only
Task Type	Diagnosis, Test, and Treatment	Treatment only
Diagnosis	Not provided	Provided
Genetic Profile	Not provided	Provided
Topic Structure	Not structured	Structured
Topic Length	Long	Relatively short
Collection	Scientific articles	Scientific abstracts
Assessment	one-step	two-step

Table 3.1: Comparison between clinical decision support and precision medicine search tasks.

3.2 Search for Clinical Decision Support

Clinical practitioners perform search tasks to look for relevant scientific articles which can assist them in making clinical decisions in regards to their patients. Clinical decision search tasks represent one category of those tasks which focuses on answering a wide range of clinical questions about providing better care to patients. For instance, clinicians might be interested in finding what the potential diagnoses are for a patient knowing a set of symptoms he or she presents, or they may be interested to know which of the lab exams and physical exams should be performed and how to recommend the right treatments for patients. Clinical decision support search tasks generally differ from other kinds of search tasks in that they target the general practice of medicine which makes these tasks ideal for simulating information seeking tasks performed by primary practitioners and physicians at intensive care units (ICUs).

In this section, we briefly describe the task and the data set we use to simulate the tasks described above. Then, we describe several reformulation approaches aiming to increase the effectiveness of those search tasks. Later in this section, we present our evaluation of the proposed approaches and provide some analysis and discuss our results.

3.2.1 Retrieval Task and Data

We use the collection and the test queries used in the TREC Clinical Decision Support (CDS) track for 2014, 2015, and 2016 tasks [104]. The collection contains over 1.25 million biomedical articles that are extracted from PubMed central. The provided test topics include 30 topics for each year (90 topics in total). Each topic represents typical information seeking task requested by clinicians in the form of a summary of patient's medical history which includes main findings, lab test, and if any previous treatments. The provided topics are categorized based on their clinical objective into three types: diagnosis, test, and treatment. Table 3.2 shows a sample of the topics we use in this work.

As discussed in Section 2.2.1.2, the TREC CDS track aims to simulate search in clinical settings. The participants in this track build their systems using various IR approaches and then submit their results (i.e., lists of potentially relevant documents) with respect to each query. Then, the submissions of all participants are judged by TREC assessors based on the relevance of those returned documents using a three-point scale with 0 representing "definitely not relevant document", 1 representing "potentially relevant document", and 2 represents "definitely relevant document". The results are then evaluated using standard IR metrics such as precision at rank k (P@k) [30] and normalized discounted cumulative gain at rank k (nDCG@k) [58].

Topic Type	Clinical Summary
Diagnosis	58-year-old woman with hypertension and obesity presents with exercise-related episodic chest pain radiating to the back.
Test	40-year-old woman with severe right arm pain and hypotension. She has no history of trauma and right arm exam reveals no significant findings.
Treatment	63-year-old heavy smoker with productive cough, shortness of breath, tachypnea, and oxygen requirement. Chest x-ray shows hyperinflation with no consolidation.

Table 3.2: Sample topics for the TREC CDS track shown as summarized clinical cases.

3.2.2 Methodology

In this section, we describe our query reformulation strategy. In Section 3.2.2.1, we introduce a method for rating clinical concepts in medical cases. Then, in Section 3.2.2.2 and Section 3.2.2.3, we discuss two reformulation approaches: applying query refinements and performing MeSH expansion, respectively.

3.2.2.1 Recommendation Model

Reformulating user queries, by removing and re-weighting some of the query terms, requires some knowledge about which of these terms are important and which are not. Thus, we develop a recommender model which can help in evaluating each concept (i.e., term or phrase) and then estimate its relevance to the search task a user aims to accomplish. For instance, the model may recommend that a concept is clinically relevant, so a suitable reformulation action is whether to keep that concept or to increase its weight in the query.

We use semantic mapping as a way to gain some knowledge about the importance of the concepts used in clinical queries. Semantic mapping refers to the process of associating terms and phrases to generic "semantic types" such as *behavior*, *clinical drug*, and *organism*. Several tools like MetaMap [12] have been developed to use advances in NLP to analyze biomedical text and to perform semantic mapping. In this work, we use MetaMap which relies on the Unified Medical Language System (UMLS)[2] to map clinical concepts to 133 distinct semantic types[3]. Table 3.3 illustrates the use of MetaMap to perform semantic mapping. As Table 3.3 illustrates, the semantic types that a concept exemplifies range from the very broad (e.g. "[Human]") to the more specific (e.g. "[Body Location or Region]"). Some of these types are irrelevant to the search task, while some may be very relevant; since a concept may map to a mix of general and specific types, it is not straightforward to use these types to improve retrieval. Therefore, we apply a "weighted voting" method by which the semantic types cast "votes" as to whether a concept is important to the search task or not.

The voting approach works as follows. First, we manually place each of the 133 semantic types obtained from MetaMap into several bags: each bag has a degree of relevance and contains a subset of semantic types. For simplicity, we assume that we have three bags: B_1 for semantic types which has the lowest relevance to clinical domain

[2] https://www.nlm.nih.gov/research/umls

[3] https://metamap.nlm.nih.gov/Docs/SemanticTypes_2013AA.txt

Concept	Semantic Mapping
Man	[Human], [Population Group], [Organism Attribute]
Shortness of breath	[Sign or Symptom], [Clinical Attribute]
Right calf tenderness	[Sign or Symptom], [Finding], [Body Location or Region]

Table 3.3: Using MetaMap to semantically map each clinical concept to several semantic types.

(e.g., temporal and geographical types), B_2 for semantic types with a higher relevance degree to clinical domain (e.g. human daily or recreational activities), and lastly, B_3 for the types with the highest relevance to clinical search domain (e.g., clinical findings, and diseases). In our selection process, 3% of semantic types (e.g., [Finding], [Injury or Poisoning], [Disease or Syndrome], and [Virus]) are assigned to B_3, 79% of the types (e.g., [Clinical Attribute], [Body Location or Region], and [Health care Activity]) are assigned to B_2, whereas the remaining 18% such as [Temporal Concept], [Geographic Area], and [Manufactured Object] are assigned to B_1. Then we assign weights to the bags based on their relevance to our search task in a way that the bag with the higher relevance degree will be given a higher weight than the other bags with lower relevance (i.e., $W(B_3) > W(B_2) > W(B_1)$). The selection/assignment procedure is performed manually by following previous studies such as Limsopatham et al. [70] and Palotti et al. [88] which provide some suggestion on which of the semantic types are more related to clinical decision making.

Finally, our recommendation model works by applying MetaMap to the topic summaries shown in Table 3.2; MetaMap automatically extracts "concepts" (i.e., terms and phrases), and for each concept c outputs a list of possible semantic types. The presence of those types in the three bags described above is then used to compute a "voting sum" for that concept as shown in the following formula:

$$Sum(c) = \sum_{i=1}^{K} W_i C_i \tag{3.1}$$

K represents the number of bags, and W_i and C_i represent the weight and the number of mapped semantic types for c in bag i, respectively. We use the result of this voting to decide what rating should be given to each concept. We define four ratings as a result of this sum which are computed as follows:

$$Rate(concept_i) = \begin{cases} 3, \text{ definitely relevant}, & \text{if } Sum(c) \geq t_3 \\ 2, \text{ potentially relevant}, & \text{if } t_3 > Sum(c) \geq t_2 \\ 1, \text{ potentially not relevant}, & \text{if } t_2 > Sum(c) > t_1 \\ 0, \text{ definitely not relevant}, & \text{if } Sum(c) \leq t_1 \end{cases} \tag{3.2}$$

In our experiments, using a partial set of the queries in Section 3.2.1 (mainly form CDS 2014) we examine several values for the bag weight W_i and the threshold t_j parameters while maximizing the retrieval effectiveness for those queries. A sample of these values is seen by setting the weights of bags such that $W_1 = -1$, $W_2 = 1$, and $W_3 = 5$, and by setting the thresholds such that $t_1 = -2$, $t_2 = 0$, and $t_3 = 5$. This allows to enforce some constraints such as a concept needs at least a sum of two negative votes to be considered “definitely not relevant” whereas it needs a sum of five positive votes to be considered “definitely relevant”. The ratings produced by the model in response to each query concept will be used by the next components in our suggested clinical search system (i.e., either query refinements or ontology-based expansion) as shown in Figure 3.1. Therefore, each component defines a set of actions to be performed in regards to each of the four ratings.

3.2.2.2 Query Refinement Model

Given a query and the ratings of all concepts in that query, we aim to apply suitable “refinements” to those concepts. We perform four types of refinements: remove a concept, reduce its weight, keep it with no change, or boost that concept’s weight.

Removing a concept

We remove each concept rated “definitely not relevant” from the target query which suggests that most of the semantic types for such concept are mapped to the types in the bag B_1. This is accomplished by excluding the removed concepts from

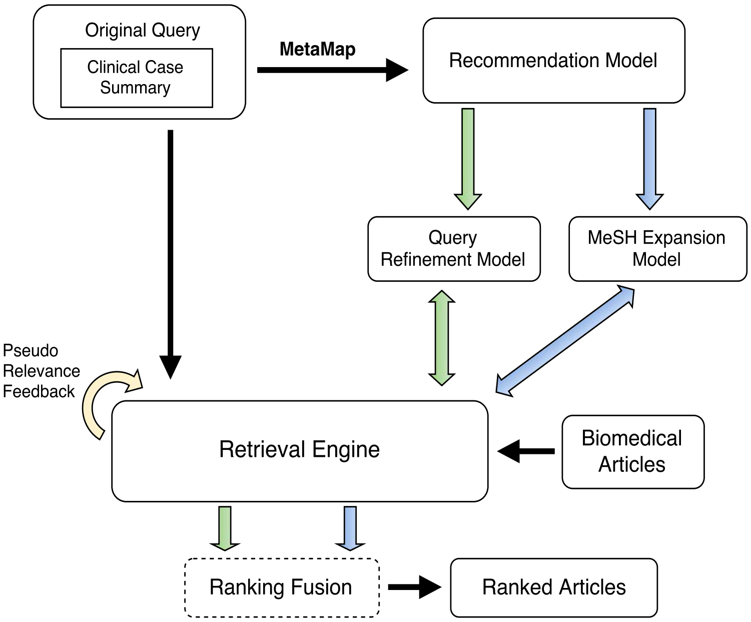


Figure 3.1: The main components of the proposed clinical search system including the recommendation model. Interactions between the system's components are illustrated by arrows.

the computation of the document scoring function that is performed by the retrieval system. Generally, IR systems rank documents by computing a score for each document with respect to a given query, and in this work, we use $InExpB2$ divergence from randomness (DFR) model [8] which computes this score as in Equation 3.3.

$$Score(d, Q) = \sum_{t \in Q} qtw \cdot w(t, d) \tag{3.3}$$

t is a term in query Q, qtw is the term weight in Q, and $w(t, d)$ is the term weight in document d computed as described in Section 2.1.1.

Reducing a concept's weight

We reduce the weight of a concept if it is rated as "potentially not relevant". This exemplifies a case when most of the mapped semantic types are shared between the two bags B_1 and B_2. We capture this refinement by modifying Equation 3.3 as:

$$Score(d, Q) = \sum_{t \in Q} qtw \cdot w(t, d) - \sum_{t \in Q_r} w_r \cdot w(t, d) \tag{3.4}$$

where Q_r represents all the terms in a query to be reduced and w_r is the reduction weight.

Keeping a concept

We keep a concept with no change if it receives "potentially relevant" rate as it suggests that the concept is relevant in answering the provided query.

Boosting a concept's weight

We increase the weight of a concept in a given query if it is rated as "definitely relevant". This exemplifies a case when most of the concept's semantic types are mapped to clinically important types, so emphasizing such concept by increasing its representation in the query is expected to have a positive impact. Also, the presence of these concepts in the introductory parts of documents (e.g., abstracts) suggests that these documents are most likely to be very relevant in answering that query. Thus, we reward those documents by applying field search mechanism such that a document's score will be boosted if the important concepts occur within the abstract field. This can be captured by modifying Equation 3.3 as:

$$Score(d, Q) = (1 - \lambda) * \left[\sum_{t \in Q} qtw \cdot w(t, d) + \sum_{t \in Q_b} w_b \cdot w(t, d) \right] + \lambda \left[\sum_{t \in Q_b} w_b \cdot w(t, d) \right] \tag{3.5}$$

where Q_b represents all the terms in a query to be boosted, w_b is the boosting weight, and λ is used to balance the abstract field's weight.

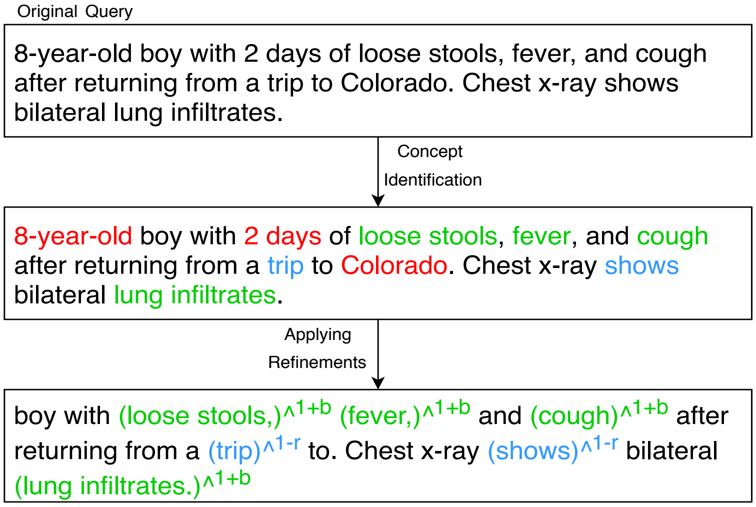

Original Query
8-year-old boy with 2 days of loose stools, fever, and cough after returning from a trip to Colorado. Chest x-ray shows bilateral lung infiltrates.
Concept Identification
8-year-old boy with 2 days of loose stools, fever, and cough after returning from a trip to Colorado. Chest x-ray shows bilateral lung infiltrates.
Applying Refinements
boy with (loose stools,)^1+b (fever,)^1+b and (cough)^1+b after returning from a (trip)^1-r to. Chest x-ray (shows)^1-r bilateral (lung infiltrates.)^1+b

Figure 3.2: Applying various refinement tasks to a clinical query. Terms that are highlighted with red, blue, black, and green represent concepts with "definitely not relevant", "potentially not relevant", "potentially relevant", and "definitely relevant" ratings, respectively.

Finally, to capture all of the described refinements above (i.e, remove and re-weight query concepts), we combine all of them by modifying Equation 3.3 as follows:

$$Score(d,Q) = (1-\lambda) * \Big[\sum_{t \in Q_\star} qtw \cdot w(t,d) + \sum_{t \in Q_b} w_b \cdot w(t,d) - \sum_{t \in Q_r} w_r \cdot w(t,d) \Big] + \lambda \Big[\sum_{t \in Q_b} w_b \cdot w(t,d) \Big] \quad (3.6)$$

where Q_* represents the query after removing irrelevant terms, Q_b and Q_r represent the terms to be boosted and those to be reduced, respectively, w_b is the boosting weight, w_r is the reduction weight, and λ is is the abstract field's weight. Figure 3.2 presents a three-step process of applying the proposed four refinement tasks to a clinical query which starts by annotating each of the concepts using the recommendation model and then applying the suitable refinement task.

3.2.2.3 MeSH Based Query Expansion Model

Having presented a method for removing and re-weighting concepts in a query, we now introduce a method for expanding those concepts rated most important using

MeSH library. Our method can identify that terminologies such as "hypertension" and "high blood pressure" refer to the same clinical concept which helps to reduce the ambiguity of clinical queries. The MeSH-based expansion is different from general expansion (known as pseudo-relevance feedback) in that it does not use feedback documents for obtaining expansion terms. Instead, it extracts medical concepts from a query and uses a MeSH hierarchical structure for expansion [135]. We rely on the ratings received from our recommendation model for each concept, and we expand only the concepts rated as "definitely relevant". Figure 3.3 shows an example of applying MeSH-based expansion on a clinical query, performed by identifying candidate concepts and then expanding each concept with related terms from MeSH tree. Generally, our approach can be applied by following the steps:

1. Concept identification: use the recommendation model to identify important concepts in a query.
2. Concept MeSH matching: use Entrez programming E-utilities [102] to link each concept with its actual MeSH entry.
3. Similarity estimation: compute semantic distance between each concept and its sibling and descendant nodes in MeSH tree.
4. Concept expansion: expand each MeSH concept with its entry terms, siblings, and descendants; however, use the estimated semantic similarity and a distance threshold δ to control the expansion.
5. Weight estimation: use PubMed collection to estimate wte_i for each expanded concept te_i.
6. Expansion combination: aggregate the expanded concepts; then, combine them with the original query.

In step 1 and step 2, we identify eligible concepts and match them with their entries in MeSH tree. In step 3, we use a semantic similarity estimation approach described in Section 3.2.2.3 to control the expansion. In other words, based on a distance threshold δ between a concept and each sibling/descendant, we either eliminate or include that sibling (or descendant) in expansion. In step 4, we use both single terms and phrases for expansion. For phrases, we apply proximity search such that

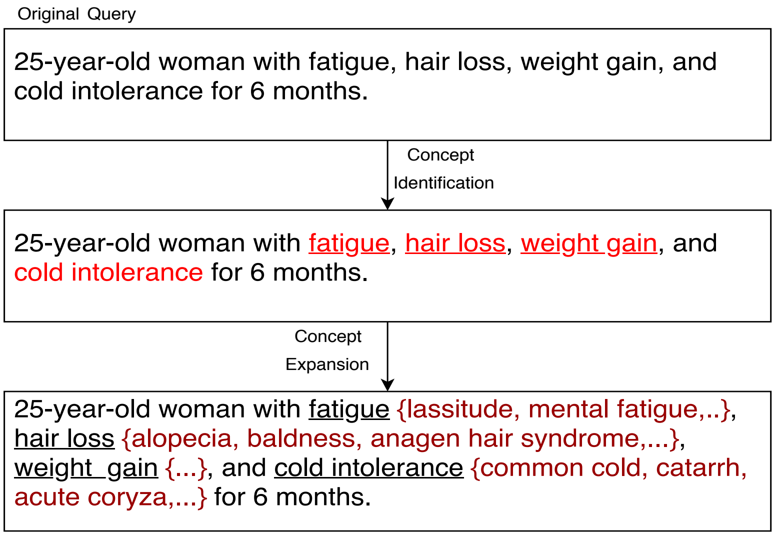

Original Query
25-year-old woman with fatigue, hair loss, weight gain, and cold intolerance for 6 months.
Concept Identification
25-year-old woman with fatigue, hair loss, weight gain, and cold intolerance for 6 months.
Concept Expansion
25-year-old woman with fatigue {lassitude, mental fatigue,..}, hair loss {alopecia, baldness, anagen hair syndrome,...}, weight gain {...}, and cold intolerance {common cold, catarrh, acute coryza,...} for 6 months.

Figure 3.3: Expanding query concepts with related terms from MeSH resource. Underlined and red-highlighted concepts represent concepts with "definitely relevant" rating.

two terms in a phrase (e.g., "gestational diabetes") are allowed to be within a distance up to n words (we set n = 16). In step 5, we use PubMed large collection (over 26 million articles) to estimate the weights of the expanded concepts. We compute the expansion weight wte_i for each concept te_i according to the following equation:

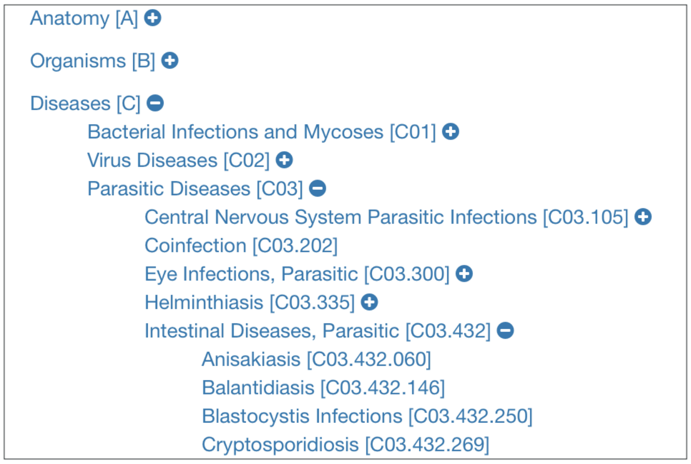
$$wte_i = \frac{\log DF(C, te_i)}{\sum_{j=1}^{K} \log DF(C, te_j)} \tag{3.7}$$

where K represents the number of expandable concepts for each MeSH concept te_i and $DF(C, te_i)$ represents the document frequency for concept te_i in PubMed collection.

Finally, in step 6, we aggregate the expanded terms into one list of size m. Then, we combine that list with the original query; accordingly, we re-define the document score as:

$$Score(d, Q) = (1 - \alpha) * \sum_{t \in Q} qtw \cdot w(t, d) + \alpha \sum_{te_i}^{m} wte_i \cdot w(te_i, d) \tag{3.8}$$

m represents the number of the expanded terms, $w(te_i, d)$ is the weight of MeSH term te_i in document d, and α is used to weight MeSH expansion with the original query.



Anatomy [A] ⊕

Organisms [B] ⊕

Diseases [C] ⊖
- Bacterial Infections and Mycoses [C01] ⊕
- Virus Diseases [C02] ⊕
- Parasitic Diseases [C03] ⊖
  - Central Nervous System Parasitic Infections [C03.105] ⊕
  - Coinfection [C03.202]
  - Eye Infections, Parasitic [C03.300] ⊕
  - Helminthiasis [C03.335] ⊕
  - Intestinal Diseases, Parasitic [C03.432] ⊖
    - Anisakiasis [C03.432.060]
    - Balantidiasis [C03.432.146]
    - Blastocystis Infections [C03.432.250]
    - Cryptosporidiosis [C03.432.269]

Figure 3.4: MeSH resource is organized in a hierarchical structure.

Estimating semantic based similarity

The MeSH vocabulary resource is organized in a hierarchical structure. As shown in Figure 3.4, the top part of that hierarchy consists of very broad medical concepts such as "Anatomy", "Mental Disorders", and "Diseases" , whereas the very bottom of the hierarchy contains very specific concepts such as "Ankle" and "Conduct Disorder" [83]. So far, MeSH resource does not provide any guidance for how the semantic distance between concepts can be measured even though such information can be beneficial when using MeSH in the applications for disambiguation and using MeSH terms for expansion. In our work, this information is needed to control expansion in order to limit broader concepts from being over-expanded with very specific concepts. Prior work [111, 135, 82, 70] try to overcome this limitation by either restricting the expansion to some branches of MeSH tree (e.g., Diseases [C]) or by only considering descendant concepts and ignore siblings. Others apply text mining techniques on several biomedical resources to estimate the relevance of the expandable concepts [70].

In this work, we rely on semantic distance to define a threshold for controlling concept expansion. Several approaches for semantic similarity measurements have been proposed (e.g., LSA [65], PMI [26], and NGD [27, 46]). But we use an approach that is based on disjunctive shared information (DiShIn) [33, 34] because it is designed to handle directed acyclic graphs (DAG) which makes it very suitable for handling the hierarchical structure of MeSH. DiShIn has been successfully applied to gene ontology and shown to be effective in estimating protein similarity as suggested by Couto et al. [33].

The way this approach works is by measuring the shared information content between two concepts which is performed by computing the average information content (IC) of their disjunctive ancestors. An ancestor concept is considered to be a common disjunctive (DCA) of two concepts only if it is the most informative shared ancestor among them or if the difference between the number of distinct paths from the two concepts to that ancestor is distinct from that of any other more informative ancestors. This definition allows DiShIn to handle parallel interpretations (i.e., multiple inheritances) when estimating the similarity between two concepts. To illustrate this property, consider the hierarchy of concepts in Figure 3.5. Based on DCA definition, DCA set for the concepts $C4$ and $C5$ contains only $C2$ whereas DCA set for $C5$ and $C6$ contains both $C2$ and $C1$. (i.e., handling parallel interpretation of semantic relation).

Shared information content between two concepts C_a and C_b would be computed according to the following formula:

$$Share(C_a, C_b) = \frac{1}{|D|} \sum_{DCA_i(C_a,C_b)\in D} IC\Big(DCA_i(C_a, C_b)\Big) \tag{3.9}$$

D is the set of disjunctive common ancestors (DCA) for C_a and C_b and IC represents the information content which is defined as:

$$IC(c) = -\log\Big(\frac{freq(c)}{maxFreq}\Big) \tag{3.10}$$

where $freq(c)$ is the frequency of concept c and its descendants and $maxFreq$ is the maximum frequency (i.e., at the root concept). Therefore, the semantic similarity between two concepts C_a and C_b can be computed as:

$$Sim(C_a, C_b) = 1 - \frac{2 * Share(C_a, C_b)}{IC(C_a) + IC(C_b)} \tag{3.11}$$

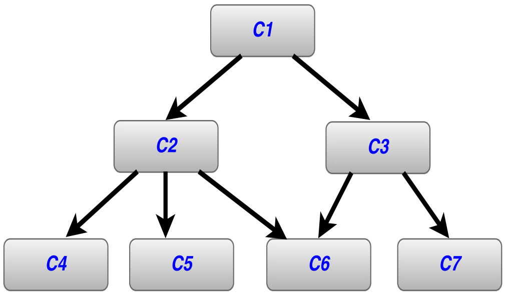


Figure 3.5: A sample hierarchical structure of concepts containing multiple inheritance.

As discussed in Section 3.2.2.3, the estimated similarity score Sim and a threshold δ (between 0.0 and 1.0) is used to formulate a decision of whether a sibling/descendant concept can be used as an expandable for a concept C.

3.2.2.4 Combining Reformulation Approaches

So far, we have discussed two reformulation approaches: one by performing refinements (i.e., removing and re-weighting concepts), while the other by expanding with terms from MeSH resource. Each method can be performed independently using the ratings received from the recommendation (i.e., applying each type of reformulations result in a ranked list of biomedical articles). In this section, we discuss how to combine the results of the two approaches into one ranked list using some ranking fusion methods.

Two types of rank fusion methods are commonly used to aggregate retrieval results from several IR systems. Score-based methods use the document score from different IR systems as a base to perform fusion; as an example, CombSUM and CombMAX use the document score from each IR system for combination by either computing a simple summation or choosing the maximum score, respectively [40]. On the hand, rank-based fusion methods rely on the rank of each document in several lists to compute a virtual document score that is ultimately used to rank that document in the final list. Generally, rank-based methods are shown to be effective in combining ranked lists as they are comparable to other types like "learning to rank" strategies which are suggested by Cormack et al. [32] and Macdonald et al. [74].

Later, in our experiments shown in Section 3.2.3.2, we will examine several rank fusion approaches. For score-based methods, we consider CombSUM which is defined as:

$$\text{CombSUM(i)} = \sum_{k=1}^{N(i)} Score_k(i) \tag{3.12}$$

where each of CombMAX(i) and CombSUM(i) represents the combined score for document i, $N(i)$ is the number of ranked lists that document i appears in, and $Score_k(i)$ is the score of document i in ranking list k. For rank-based methods, we will exam two approaches Reciprocal Rank Fusion (RRF) [32, 131] and Inverse Square Rank (ISR) [82]. They are defined as follows:

$$RRF(i) = \sum_{k=1}^{N(i)} \frac{1}{60 + R_k(i)} \tag{3.13}$$

$$ISR(i) = N(i) * \sum_{k=1}^{N(i)} \frac{1}{R_k(i)^2} \tag{3.14}$$

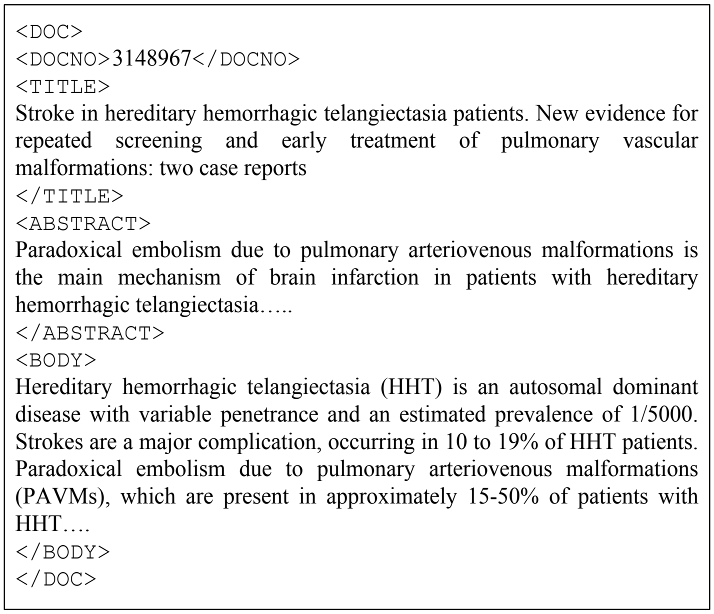
where each of $RRF(i)$ and $ISR(i)$ represents combined score for document i, $N(i)$ is the number of ranked lists that document i appears in, and R_k is the rank of document i in ranking list k.

3.2.3 Evaluation

In this section, we describe our evaluation of the two proposed reformulation methods. We start by defining our experimental designs and settings. Then, we present our results and finally, provide some analysis and discussion.

3.2.3.1 Experimental Settings

We use the topic sets for the TREC CDS track 2014, 2015, and 2016 as described in Section 3.2.1. The topic sets consist of 60 synthetic cases for general practice patients that were used in 2014 and 2015 tasks (30 cases for each year) in addition to actual



```
<DOC>
<DOCNO>3148967</DOCNO>
<TITLE>
Stroke in hereditary hemorrhagic telangiectasia patients. New evidence for repeated screening and early treatment of pulmonary vascular malformations: two case reports
</TITLE>
<ABSTRACT>
Paradoxical embolism due to pulmonary arteriovenous malformations is the main mechanism of brain infarction in patients with hereditary hemorrhagic telangiectasia…..
</ABSTRACT>
<BODY>
Hereditary hemorrhagic telangiectasia (HHT) is an autosomal dominant disease with variable penetrance and an estimated prevalence of 1/5000. Strokes are a major complication, occurring in 10 to 19% of HHT patients. Paradoxical embolism due to pulmonary arteriovenous malformations (PAVMs), which are present in approximately 15-50% of patients with HHT….
</BODY>
</DOC>
```

Figure 3.6: An example of a processed biomedical article which contains the most informative fields.

30 cases taken from admission notes for ICU (Intensive Care Unit) patients which were used in 2016 task. We index the target collection using Terrier [85] by applying default indexing settings such as Porter stemmer and the default stopword list. Each document (i.e., article) is fully indexed by extracting all the textual content including title, abstract, paragraphs, subheadings, and figure labels. Ultimately, each document is represented by the four fields: `<DOCNO>`, `<TITLE>`, `<ABSTRACT>`, and `<BODY>`. Figure 3.6 illustrates a sample biomedical article after extracting the four fields.

We perform retrieval using a baseline that relies on $InExpB2$ DFR retrieval model. Also, we apply pseudo relevance feedback (PRF) query expansion that is based on KL divergence [8]. We set expansion parameters such that we expand each query with 110 terms from the top 5 documents as suggested in [6].

For MeSH expansion, we follow the description of DiShIn [33] to apply semantic similarity estimation. Specifically, we parse the original MeSH resource file (contains 27,883 concepts) and create a filtered set of attributes for each concept (e.g., concept heading, terms, and tree IDs) which allows to load and manipulate MeSH data during runtime efficiently. Due to the way the concepts are arranged in MeSH tree using alphabetic IDs (e.g., C02.440.420 and C02.440.440 represent tree ids for Hepatitis A and C, respectively and both have the same parent Hepatitis with id C02.440), the estimation of semantic similarity using disjunctive common ancestors is even more time-efficient than the original DiShIn presented in [33].

For parameter selection, we set the parameters λ, α, δ, w_b, and w_r in our reformulation approaches described in Section 3.2.2 using six-fold cross-validation with P10 metric. This means for each topic set; we use six-fold cross-validation to train these parameters such that in each fold we train with 25 topics and test with 5. We select parameter values that give the highest average performance across the six testing folds. Lastly, we perform additional experiments to examine the effectiveness of combining search results from different reformulation approaches using rank fusion methods such as CombSUM, CombRRF, and CombISR. The results of all these experiments will be discussed in the next section.

3.2.3.2 Results and Analysis

We apply the two query reformulation tasks (i.e., query refinements and MeSH expansion) discussed in Section 3.2.2 to the three topic sets 2014, 2015, and 2016, and we compare our results to a baseline system that uses $InExpB2$ model and KL query expansion. Table 3.4 summarizes the results of our experiments using P@5, P@10, and nDCG@10 [30] as main evaluation metrics. Also, Figure 3.7 shows the precision-recall graph which compares the different reformulation approaches and the baseline by plotting the relationship between precision and recall. Lastly, Figure 3.8 presents a performance comparison at query-level between each method and the baseline using P@10 as the main comparison metric.

From Table 3.4, Figure 3.7, and Figure 3.8, one can see that applying query refinements by removing and re-weighting some of the query terms can lead to significant performance improvement. More specifically, for the two topic sets CDS 2014 and

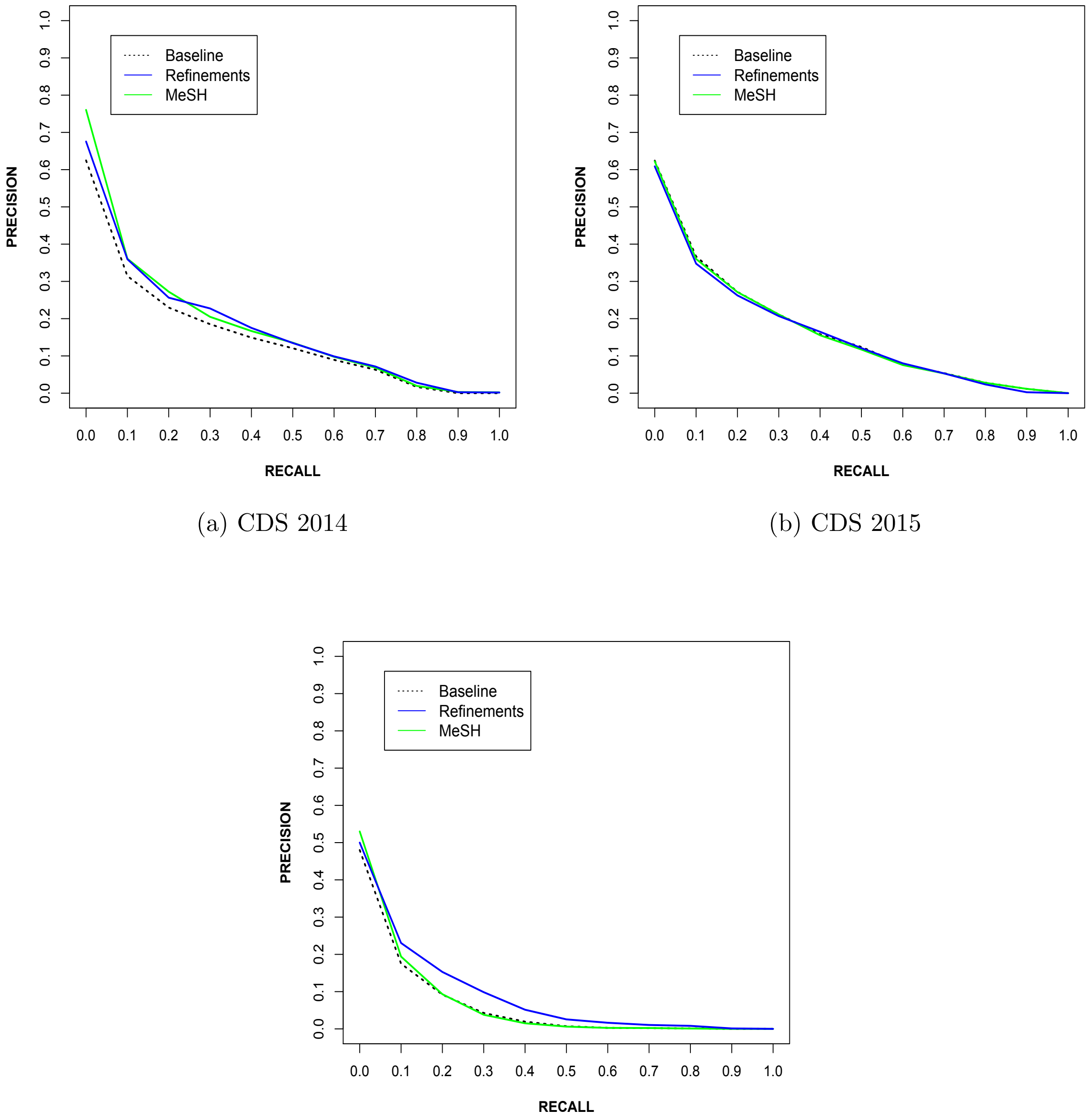
CDS 2014						
Approach	**P@5**	**%Δ**	**P@10**	**%Δ**	**nDCG@10**	**%Δ**
Baseline	0.3800	–	0.3500	–	0.2899	–
Refinements (cv)	**0.4667****	+22.8%	**0.4233****	+20.9%	0.3406*	+17.4%
MeSH expansion (cv)	0.4200*	+10.5%	0.3900	+11.4%	**0.3511***	+21.1%
Refinements (best)	**0.4667****	+22.8%	**0.5000****	+42.8%	**0.3922****	+35.2%
MeSH expansion (best)	0.4333*	+14.0%	0.4533**	+29.5%	0.3749**	+29.3%

CDS 2015						
Approach	**P@5**	**%Δ**	**P@10**	**%Δ**	**nDCG@10**	**%Δ**
Baseline	**0.4467**	–	**0.4100**	–	0.2706	–
Refinements (cv)	0.4066	-9.8%	0.3967	-3.2%	**0.2796**	+3.3%
MeSH expansion (cv)	0.4400	-1.5%	0.4033	-1.6%	0.2682	-1.0%
Refinements (best)	**0.4733**	+6.0%	**0.4766****	+16.2%	**0.3166****	+16.9%
MeSH expansion (best)	0.4333	-3.0%	0.4566**	+11.3%	0.2946**	+8.8%

CDS 2016						
Approach	**P@5**	**%Δ**	**P@10**	**%Δ**	**nDCG@10**	**%Δ**
Baseline	0.2800	–	0.2767	–	0.1964	–
Refinements (cv)	0.3267*	+16.7%	**0.3133***	+13.2%	0.2100	+7.0%
MeSH expansion (cv)	**0.3467***	+23.8%	0.3033	+9.6%	**0.2131**	+8.5%
Refinements (best)	**0.3600****	+28.6%	**0.3998****	+45.5%	**0.2517****	+28.15%
MeSH expansion (best)	0.3533*	+26.17%	0.3800**	+37.3%	0.2463**	+25.4%

Table 3.4: Performance results of applying two reformulation approaches on TREC CDS topics. Statistical significance (paired t-test) over the baseline is denoted by * for $p < 0.05$ and ** for $p < 0.01$. Bold-faced values represent the approaches with maximum performance.

CDS 2016, we observe a substantial performance increase over the baseline climbing up to 22.8% increase with P@5, 20.9% with P@10, and 17.4% with nDCG@10 for 2014 topics whereas for 2016 topics, the performance increase is 16.7% with P@5, 13.3% with P@10, and 7.0% with nDCG@10 metric. Also, looking at Figure 3.8(a), we can see that applying query refinements is consistent on the two topic sets (topics 1–30, and topics 61–90) resulting in a positive impact for about half of the topics in each topic set. In addition, by performing paired one-sided t-test on those results, this improvement is shown to be statistically significant for both 2014 and 2016 topics (i.e., $p < 0.01$ for P@5 and P@10 metrics, and $p < 0.05$ for nDCG@10 considering CDS 2014 topics whereas for 2016 topics, $p < 0.05$ for P@5 and P@10).



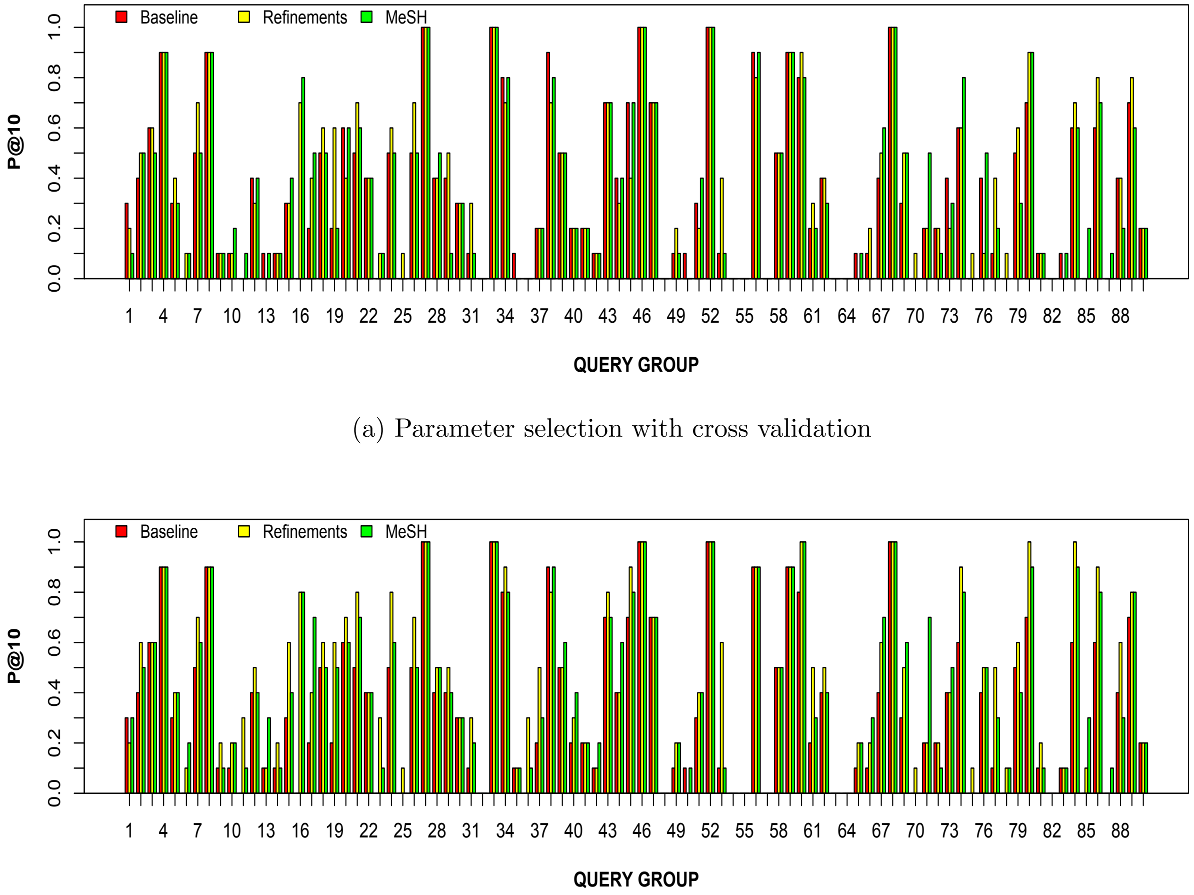
(a) CDS 2014

(b) CDS 2015

(c) CDS 2016

Figure 3.7: Comparison of Precision-Recall for query refinements, MeSH expansions, and the baseline for the three topic sets CDS 2014, CDS 2015, and CDS 2016.

Moreover, our refinement approach has an upper bound (i.e., best as indicated in Table 3.4) which can achieve over 45% performance increase when many training



(a) Parameter selection with cross validation

(b) Parameter selection with best settings

Figure 3.8: Per query performance comparison with P@10 for each of the reformulation approaches and the baseline. Two cases are considered for selecting reformulation parameters.

samples are available and when selecting parameters using approaches such as feature based learning approaches. On the other hand, for the topics used in 2015 CDS task, the performance improvement is lower than both 2014 and 2016, and in fact looking at the case of setting parameters with cross-validation, only nDCG@10 shows a slight increase over our baseline. Later in this section, we will address how the impact of the proposed refinements can vary based on the used topic set.

Likewise, applying our MeSH expansion approach led to a noticeable improvement over the baseline. In particular, for 2014 topics, the performance increase is shown to be 10.5% with P@5 metric, 11.4% with P@10, and 21.1% with nDCG@10. Also, for 2016 topics, the improvement over the baseline is 23.8% with P@5, 9.6% with P@10, and 8.5% with nDCG@10. Moreover, as indicated by Figure 3.8(a), we see that the positive impact of this approach seems to be present and shown to result in improving at least a third of the topics for each set. Similar to the results discussed above, our proposed MeSH expansion has the potential to be enhanced by reaching an upper bound of over 35% performance increase. Another observation is that the effectiveness of applying MeSH expansion to 2015 topics is shown to be lower than the other topic sets as discussed for the case of applying query refinements as indicated above.

To understand why the effect of those approaches may vary from one topic set to another, we perform further analysis by looking into the differences between the topic sets and to how the reformulations are applied to each set individually. One clear difference is the length (in terms) among the topic sets as it has been shown that topics for 2015 task tend to be very short (average query length is 15.8 terms) compared to 2014 and 2016 sets (average query length is 19.6 terms for 2014 and 23.33 terms for 2016). The length of queries correlates very well with the average retrieval performance of the baseline used to answer each topic set (i.e., before applying reformulations), suggesting that there is an observed negative impact on the performance due to longer queries (i.e., long queries tend to be more difficult than short queries), which is also suggested by He et al. [51] and others [89, 103]. Moreover, we observe another difference between the sets by looking into the topics after applying reformulations. Our results show that among those sets, the topics for 2015 are confirmed to contain very few terms that are categorized as either "not relevant" or "potentially not relevant" which suggests that these topics are written more concisely, making the important, and the potentially important clinical concepts to be well represented. Therefore, these topics are easier to retrieve even without applying reformulations (in fact, applying reformulation might

Task	Length	Clarity	Scope	AvPMI	AvICTF	AvNGD	WIG	QF	SCQ
2014	19.6	8.6	0.20	6.0	12.8	0.72	6.0	0.22	33.5
2015	15.8	9.4	0.22	6.2	13.3	0.70	6.3	0.23	35.1
2016	23.3	8.9	0.19	5.8	11.1	0.86	5.2	0.20	35.3

Table 3.5: Estimation of query difficulty for TREC CDS topics. underlined values represent statistics with higher performance prediction (i.e., lowest difficulty).

affect those topics negatively).

Lastly, to confirm that there is indeed a difference between topic sets in regards to difficulty and easiness, we consider computing query performance predictors for each topic set. These predictors evaluate the easiness of each query such that when a predictor's value is high, the retrieval performance is estimated to be high and, therefore, a query is assumed to be easy, whereas when the value is low, the query is assumed to be difficult. In Table 3.5, we show that the topics for 2015 have higher values (i.e., easier) than both 2014 and 2016 with six predictors out of the eight predictors (clarity [51], scope [51], AvgPMI [50], AvgICTF [51], AvgNGD [27], WIG [134], QF [134], and SCQ [133]) used in this analysis.

Combining Reformulation Approaches

The last set of experiments we conduct is by applying several rank fusion methods to combine the two reformulation approaches, query refinements and MeSH expansion, and generate a single list of search results for each given query. In our experiments, we use three rank fusion methods, CombSUM, CombRRF, and CombISR, described in Section 3.2.2.4. The results of our evaluation are presented in Table 3.6 with P@5, P@10, and nDCG@10 as main evaluation measurements.

Comparing our results in Table 3.6 to those discussed previously in Table 3.4, we see that applying rank fusion methods has led to more inconsistent results, improving retrieval effectiveness for one metric while degrading the effectiveness for the others. For instance, when combining search lists generated with best settings for the two

CDS 2014						
Approach	**P@5**	**%Δ**	**P@10**	**%Δ**	**nDCG@10**	**%Δ**
Baseline	0.3800	–	0.3500	–	0.2899	–
CombSUM Fusion (cv)	**0.4533**	+19.3%	0.4233	+20.9%	0.3478	+20.0%
CombRRF Fusion (cv)	0.4467	+17.5%	0.4200	+20.0%	0.3453	+19.1%
CombISR Fusion (cv)	0.4333	+14.0%	**0.4333**	+23.8%	**0.3580**	+23.5%
CombSUM Fusion (best)	**0.4933**	+29.8%	0.4311	+23.2%	**0.3747**	+29.3%
CombRRF Fusion (best)	0.4733	+24.6%	**0.4433**	+26.7%	0.3740	+29.0%
CombISR Fusion (best)	0.4533	+19.3%	0.4333	+23.8%	0.3678	+26.9%

CDS 2015						
Approach	**P@5**	**%Δ**	**P@10**	**%Δ**	**nDCG@10**	**%Δ**
Baseline	**0.4467**	–	0.4100	–	0.2706	–
CombSUM Fusion (cv)	0.4333	-3.0%	0.4067	-1.0%	0.2675	-1.1%
CombRRF Fusion (cv)	0.4333	-3.0%	**0.4167**	+1.0%	**0.2747**	+1.5%
CombISR Fusion (cv)	0.4200	-6.0%	0.4033	-1.6%	0.2734	+1.0%
CombSUM Fusion (best)	0.4600	+3.0%	0.4400	+7.3%	0.2967	+9.6%
CombRRF Fusion (best)	**0.4667**	+4.5%	**0.4467**	+9.0%	**0.2992**	+10.6%
CombISR Fusion (best)	0.4333	-3.0%	**0.4467**	+9.0%	0.2974	+9.9%

CDS 2016						
Approach	**P@5**	**%Δ**	**P@10**	**%Δ**	**nDCG@10**	**%Δ**
Baseline	0.2800	–	0.2767	–	0.1964	–
CombSUM Fusion (cv)	0.3267	+16.7%	0.3033	+9.6%	0.2050	+4.4%
CombRRF Fusion (cv)	0.3133	+11.9%	0.3033	+9.6%	0.2060	+4.9%
CombISR Fusion (cv)	**0.3533**	+26.2%	**0.3100**	+12.0%	**0.2096**	+6.7%
CombSUM Fusion (best)	0.3600	+28.6%	0.3367	+21.7%	0.2306	+17.4%
CombRRF Fusion (best)	**0.3667**	+31.0%	0.3200	+15.6%	0.2285	+16.3%
CombISR Fusion (best)	0.3467	+23.8%	**0.3633**	+31.3%	**0.2382**	+21.2%

Table 3.6: Performance results of applying three rank fusion approaches CombSUM, CombRRF, and CombISR on TREC CDS topics. Bold-faced values represent the approaches with maximum performance.

types of reformulations, we see that in many cases, the effectiveness of P@5 seems to be slightly improved compared to if each single reformulation approach is used. In contrast, for P@10 and nDCG@10, we observe a significant performance reduction reaching up to 11.34% compared to if one of the two reformulation approaches is used by its own. One interesting observation is that even when the effectiveness of ranking fusion is not the highest among each individual approach, the resulting performance tends to be at an average point of the performance of the two individual reformulations. This actually suggests in the worst case, the resulting performance of the rank fusion methods will not be worse than the minimum of the two reformulation approaches.

3.3 Precision Medicine Search Tasks

In Section 3.2, we introduced and discussed clinical decision search tasks. Also, we described two approaches for applying query reformulations on these search tasks: query refinements and MeSH expansions. In this section, we provide a brief introduction of search tasks for precision medicine. Furthermore, we examine and study the application of reformulation approaches to this category of search tasks as well.

Generally, search tasks for precision medicine differ from clinical decision support tasks in that they focus on answering clinical questions that are more specific to certain medical conditions or a certain group of patients while taking into considerations some individualized aspects of the patients such as genetic, environmental, and social profiles. Particularly, in this work, we consider only the tasks which focus on assisting clinicians (mainly oncologists) to search and retrieve relevant biomedical articles which help in providing better treatments to patients diagnosed with cancer. In the next few sections, we describe the dataset we use for this task, and we discuss how to extend our proposed approaches to those tasks.

3.3.1 Retrieval Task and Data

We use the collection and the test queries provided in the TREC Precision Medicine (PM) track for 2017 task [94, 112]. The collection contains over 27 million scientific abstracts which represent PubMed large collection. The topics used in this task were prepared by oncologists at the University of Texas MD Anderson Cancer center and consisted of 30 structured clinical cases for patients diagnosed with different types of cancer. Each case describes very precise information about a given patient which includes demographic information (such as age and sex), type of cancer, one or more gene variant, and any other significant information. The focus of the task is to

ID	Patient Case
Case 1	*Disease:* Liposarcoma *Gene:* CDK4 Amplification *Demographic:* 38-year-old male *Other:* GERD
Case 4	*Disease:* Breast cancer *Gene:* FGFR1 Amplification, PTEN (Q171) *Demographic:* 67-year-old female *Other:* Depression, Hypertension, Heart Disease
Case 12	*Disease:* Colon cancer *Gene:* BRAF (V600E) *Demographic:* 35-year-old male *Other:* None

Table 3.7: Example topics from the TREC Precision Medicine Track shown as structured clinical cases.

find biomedical articles which can be relevant in assisting oncologists in treating their patients. Table 3.7 shows a sample of the topics we use in this work.

Similar to the other TREC tracks, the participants in this track were asked to examine a variety of IR approaches and then submit their results (i.e., lists of potentially relevant documents) concerning each query. Then, following typical TREC procedures, the submissions of all participants were judged by TREC assessors based on the relevance of those returned documents. However, instead of the one-step assessment process that is performed in TREC CDS track (described in Section 3.2.1), the track's organizers have chosen to perform two-tiered assessment [113, 94]. The process was applied such that first, a manual assessment was made by the human assessors based on several categories for each result (e.g., is it an animal or human PM article? does it contain the exact or general disease?), then a relevance score was assigned to the result based on its categorization. Finally, each document will have either 0,1, and 2 such that 0, the lowest score, represents "definitely not relevant document" whereas 2, the highest, is representing "definitely relevant document".

3.3.2 Extending Query Reformulations on Precision Medicine

Before applying any of the proposed reformulation approaches in Section 3.2.2, we want to discuss some of the main differences between the two search tasks. We assume that this difference between the two tasks may affect the deployment of the reformulation approaches from one task to another.

One fundamental difference between the two tasks is the type of information need requested by healthcare practitioners. One one hand, in clinical decision search tasks, we assume that the clinicians would be interested in a wide range of clinical questions such as the potential diagnosis for a given patient knowing the set of symptoms he or she exemplifies, the lab and physical testings that clinicians need to perform to rule out some possible conditions, and the available procedures and medical operations for treatment. In all of these questions, we expect that part of the performed search task is to find the right diagnoses of the patient. On the other hand, in search tasks for precision medicine, we assume that the given clinical question would be very precise on finding the available treatment plans for the given patient. Also, we suppose that clinicians would know in advance the right diagnosis for the patient as a result from previous information seeking tasks (we may think that clinical decision support search tasks were completed at an early stage to assist in finding diagnoses, then precision medicine search tasks are performed at a later stage to seek treatments).

In addition, due to the fact that precision medicine is specific to a particular group of patients (i.e., with cancer diseases), we expect the provided information need in regards to those patients to share some common attributes (e.g., age, sex, cancer type, and gene variant), leading to the possibility of writing patients' medical cases in more structured way and a shorter form compared to the cases for general medicine. This consequently will also result in changing some settings and assumptions about the task itself such as the way to perform evaluations, and how to decide whether a given article is relevant or not.

To summarize, we may be able to extend some of the previously proposed approaches for general clinical decision search tasks; however, some alterations and assumptions should be considered to make these approaches suitable for search tasks in precision medicine. In the next section, we explain how we modify our methods and extends them to the new search task.

3.3.3 Methodology

Having presented some discussions about the main differences between the two clinical search tasks, in this section, we describe how to extend some of the previously proposed reformulation strategies. We mainly focus on query refinements and applying ontology expansions.

3.3.3.1 Query Refinements

Similar to what we described in Section 3.2.2.2, we focus on applying a set of refinement tasks to enhance the representation of clinical queries by highlighting the most relevant content in those queries. For a given concept in a query, we consider applying three refinement tasks: reducing a concept's weight, keeping a concept, and boosting a concept's weight (due to the short length of the queries, we do not consider removing any of their concepts).

Reducing a concept's weight

We reduce the weights of the concepts which exemplify low importance to the clinical search task. In particular, for this refinement task, we consider the concepts which describe demographic information (e.g., age and sex) as they are expected to have the lower contribution in finding treatment options for cancer patients. We can identify concepts related to the age and the sex of the patients by either extracting them directly from `<demographic>` field or applying semantic mapping with MetaMap in the case the provided query is not structured (i.e., age is extracted from [Temporal Concept] mapping whereas sex is extracted from [Organism Attribute]).

Boosting a concept's weight

We increase the weights of all the concepts that are expected to be essential in finding the potential treatment plans for cancer patients. More specifically, we consider both the type of cancer disease, and the gene variant to be significantly important, so we boost the weights of their terms in the original query. Similar to what we described above, we identify diseases and genes in clinical cases by extracting them from their corresponding fields `<disease>` and `<gene>`, respectively, or by applying semantic mapping with MetaMap (disease is extracted from [Neoplastic Process] semantic type, while gene is extracted from [Cell or Molecular Dysfunction]).

Keeping a concept

The concepts that are included in the `<other>` field of a precision medicine case can describe several aspects of a patient's health such as any additional conditions, symptoms, and the status of the current cancer stage. We assume that these concepts might be potentially relevant in seeking the right cancer treatments; however, due to the variety of information which can be included in the `<other>` field, it might not be easy to decide which are more relevant and which are not. Therefore, for simplicity, we choose to keep all of these concepts without applying neither boosting nor weight reduction.

Finally, to capture all of these refinements in our ranking model, we slightly modify Equation 3.6 in Section 3.2.2.2 as follows:

$$Score(d,Q) = \left[\sum_{t \in Q} qtw \cdot w(t,d) + w_{b_1} \cdot \sum_{t \in Q_{b_1}} w(t,d) + \sum_{t \in Q_{b_2}} w_{b_2} \cdot w(t,d) - \sum_{t \in Q_r} w_r \cdot w(t,d)\right] \tag{3.15}$$

where Q represents the set of all query terms, Q_{b1} and Q_{b2} represent terms associated with diseases and genes, respectively, Q_r represents the terms associated with demographic attributes, w_{b_1} and w_{b_2} are boosting weight, and w_r is the reduction weight. In Figure 3.9, we show a three-step process for applying the proposed refinement tasks

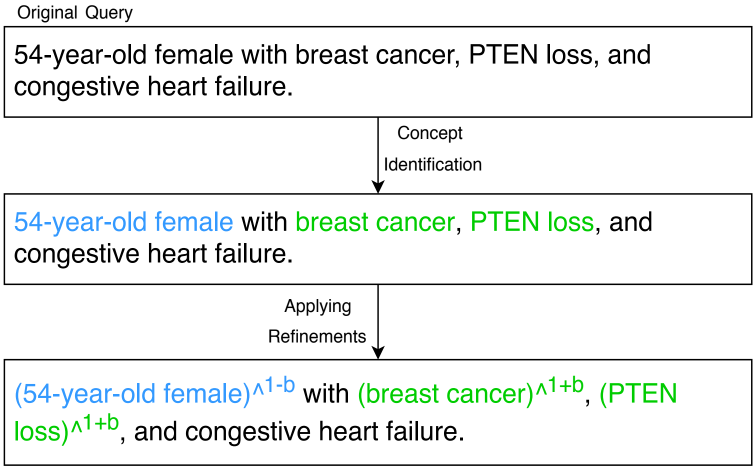

Original Query
54-year-old female with breast cancer, PTEN loss, and congestive heart failure.
Concept Identification
54-year-old female with breast cancer, PTEN loss, and congestive heart failure.
Applying Refinements
(54-year-old female)^1-b with (breast cancer)^1+b, (PTEN loss)^1+b, and congestive heart failure.

Figure 3.9: Applying various refinement tasks to a query for precision medicine. Terms that are highlighted with blue represent demographic information whereas terms highlighted with green represent disease and gene attributes.

to a sample clinical query which starts by identifying each of the concepts and then applying the suitable refinement tasks.

3.3.3.2 Ontology Based Expansions

We consider two methods for reducing the ambiguity of concepts in clinical queries. First, we expand each cancer disease with its related names from MeSH library consisting of over 27 thousand concepts. Then, we expand each gene variant with its other names from Genetics Home Reference[4] resource which contains over 1400 referenced genes, which are sorted by the alphabetic order of their names.

For MeSH expansion, we follow our approaches described in Section 3.2.2.3 to expand and re-weight each of the identified diseases. Also, to limit the effect of query drift, we only consider the condition identified in `<disease>` field. Likewise, for gene expansion, we follow the same procedure, but we replace MeSH with Genetics Home Reference resource, and we expand all the genes in `<gene>` field. Figure 3.10 and Figure 3.11 illustrate the process of extracting diseases and genes from clinical cases and then

[4] https://ghr.nlm.nih.gov/gene

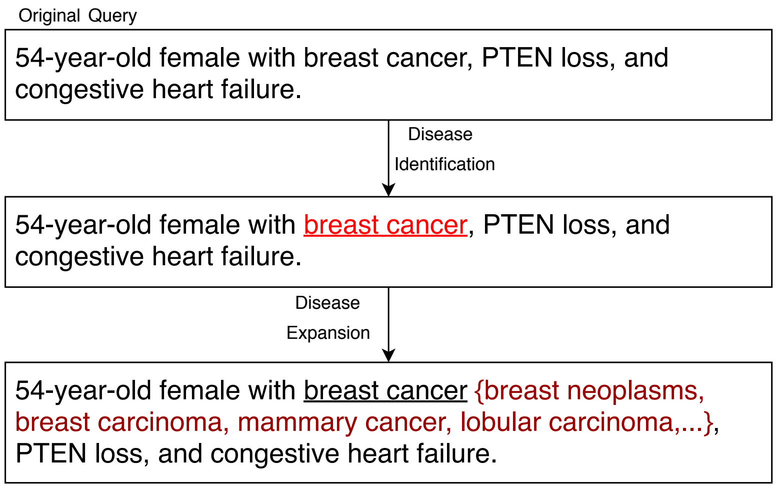

Original Query
54-year-old female with breast cancer, PTEN loss, and congestive heart failure.
Disease Identification
54-year-old female with breast cancer, PTEN loss, and congestive heart failure.
Disease Expansion
54-year-old female with breast cancer {breast neoplasms, breast carcinoma, mammary cancer, lobular carcinoma,...}, PTEN loss, and congestive heart failure.

Figure 3.10: Expanding each cancer disease with its related name from MeSH tree resource.

expanding them with their related terms from MeSH and Genetics Home Reference, respectively.

3.3.4 Evaluation

In this section, we describe our evaluation of applying reformulation approaches on search tasks for precision medicine. First, we discuss our experimental designs and settings. Then, we present our results and provide some analysis and discussion.

3.3.4.1 Experimental Settings

The target collection used for this task consists of a recent snapshot of PubMed abstracts (about 27 million titles) as well as a sample of the abstracts from AACR and ASCO proceedings (over 70 thousands titles). In our indexing, we performed an additional step for cleaning the data set by removing all the empty documents in the collection (i.e., documents with title but no abstract text). This result in a total of 16,857,746 abstracts. Then, as shown in Figure 3.12, for each XML document, we extracted the three fields `<DOCNO>`, `<TITLE>`, and `<ABSTRACT>` from their corresponding tags and created a ready-to-index TREC document. Lastly, we used Terrier to build

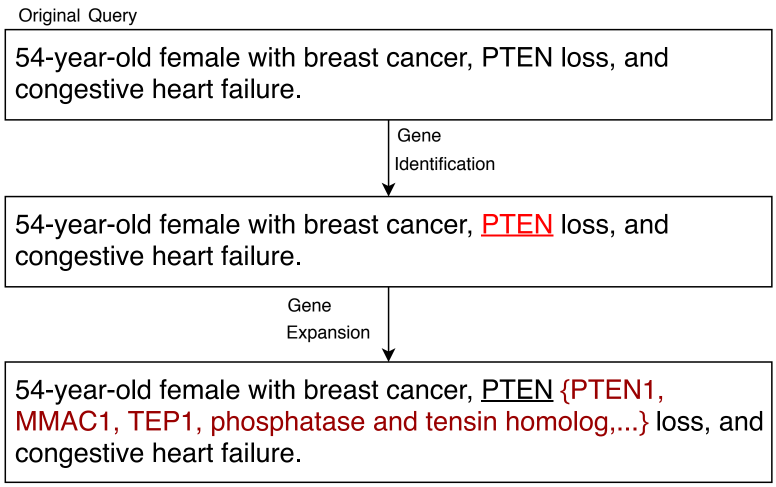
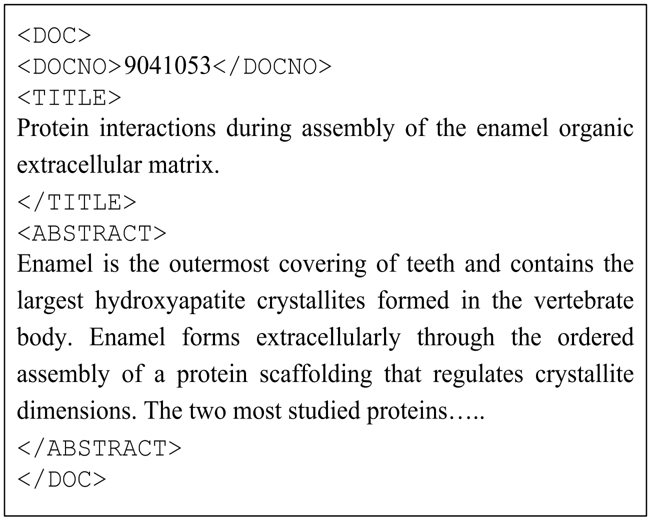

Original Query
54-year-old female with breast cancer, PTEN loss, and congestive heart failure.
Gene Identification
54-year-old female with breast cancer, PTEN loss, and congestive heart failure.
Gene Expansion
54-year-old female with breast cancer, PTEN {PTEN1, MMAC1, TEP1, phosphatase and tensin homolog,...} loss, and congestive heart failure.

Figure 3.11: Expanding each gene variant with its other names from Genetics Home Reference library.

an index for all the resulted documents and we apply default indexing settings such as Porter stemming algorithm and default stopword list.

For retrieval, the strategy we follow is similar to what is discussed in Section 3.2.3.1. Specifically, we use a baseline that relies on $InExpB2$ DFR retrieval model, but we do not use pseudo relevance feedback (PRF) as it has shown to degrade the retrieval performance. Furthermore, we apply MeSH disambiguation by expanding with only cancer diseases in each of the queries. Likewise, we expand each of the gene names with their other names using Genetics Home Reference resource that is provided by NLM.

We set the parameters in each of the reformulation approaches used in our experiments by applying six-fold validation with P10. In each fold, we train with 25 topics and test with 5, and then, we select parameter values that give the highest average performance across the six testing folds. Also, to show the upper bound performance of our approaches, we set parameter values to their best settings such that retrieval effectiveness is maximized.



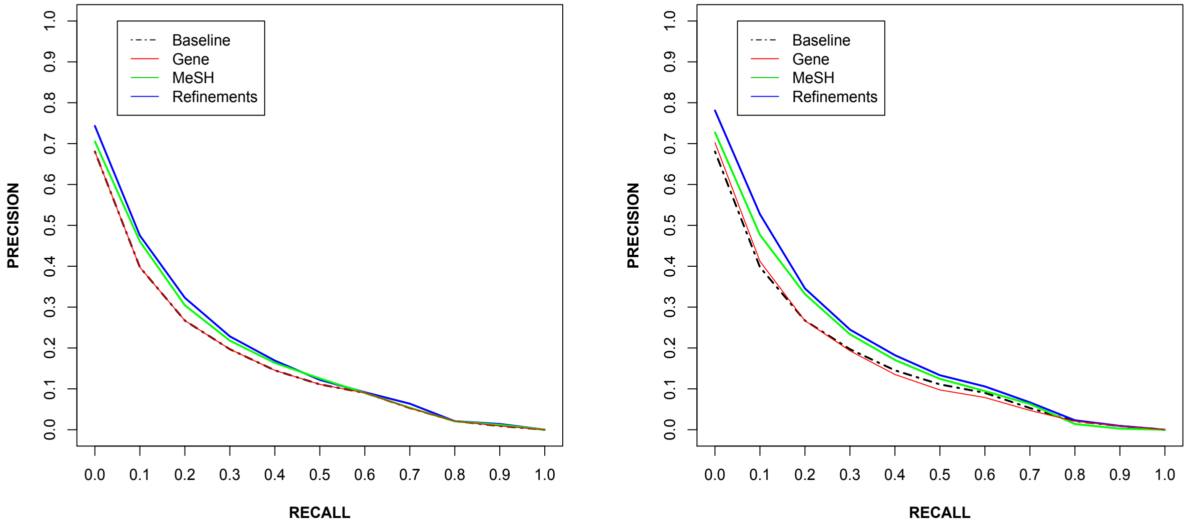
```
<DOC>
<DOCNO>9041053</DOCNO>
<TITLE>
Protein interactions during assembly of the enamel organic extracellular matrix.
</TITLE>
<ABSTRACT>
Enamel is the outermost covering of teeth and contains the largest hydroxyapatite crystallites formed in the vertebrate body. Enamel forms extracellularly through the ordered assembly of a protein scaffolding that regulates crystallite dimensions. The two most studied proteins…..
</ABSTRACT>
</DOC>
```

Figure 3.12: A Sample biomedical article after extracting the three fields `<DOCNO>`, `<TITLE>`, and `<ABSTRACT>`.

3.3.4.2 Results and Analysis

We apply each of the reformulation tasks discussed in Section 3.3.3 (i.e., query refinements, and ontology expansions) to the 30 topics for the TREC PM track. We compare the retrieval effectiveness of these approaches to a baseline search system which only uses $InExpB2$ ranking model. Also, we examine four cases for applying reformulation approaches: applying only weight reduction of demographic concepts, applying all refinements (reduction + boosting), applying MeSH expansion, and applying gene expansion. Table 3.8 summarizes the results of our experiments with P@5, P@10, and nDCG@10 [30] as the main evaluation metrics. In addition, Figure 3.13 shows the precision-recall graph to evaluate the proposed approaches by plotting the relationship between precision and recall whereas Figure 3.14 presents per query performance comparison with P@10 metric for all reformulation approaches.

Looking at the results from Table 3.8, we see a substantial performance effect when applying a variety of reformulation approaches. In particular, applying query refinements, by boosting the weights of important terms and reducing the weights of others expected to be less important, is shown to result in noticeable impact over the



(a) Cross validation

(b) Best settings

Figure 3.13: Comparison of Precision-Recall (averaged among 30 queries) for several reformulation approaches and the baseline is shown for selecting parameters with cross validation (left) and choosing the best settings (right).

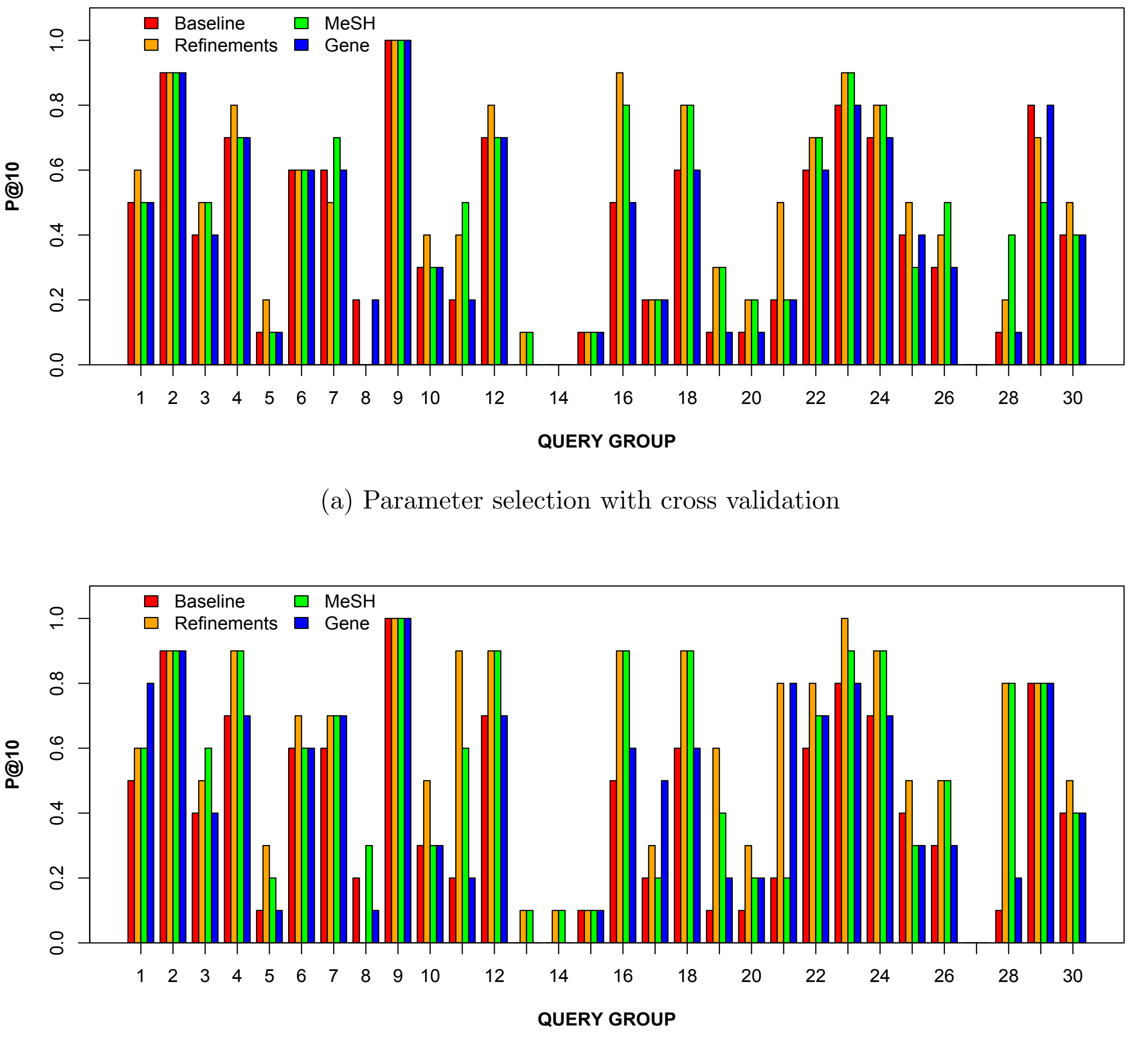
Approach	**P@5**	%Δ	**P@10**	%Δ	**nDCG@10**	%Δ
Baseline	0.4133	-	0.4033	-	0.3812	-
Weight reduction (cv)	0.4866*	+17.7%	0.4466*	+10.7%	0.4309*	+13.0%
Query refinements (cv)	**0.5066***	+22.6%	**0.4833****	+19.8%	**0.4575****	+20.0%
MeSH expansion (cv)	0.4600	+11.3%	0.4567*	+13.2%	0.4252*	+11.5%
Gene expansion (cv)	0.4066	-1.6%	0.4033	+0.0%	0.3836	+0.6%
Weight reduction (best)	0.4800*	+16.1%	0.4500*	+11.6%	0.4340*	+13.8%
Query refinements (best)	**0.5467****	+32.3%	**0.5933****	+47.1%	**0.5429****	+42.4%
MeSH expansion (best)	0.5133**	+24.2%	0.5333**	+32.2%	0.4781**	+25.4%
Gene expansion (best)	0.4333	+4.8%	0.4566*	+13.2%	0.4208*	+10.4%

Table 3.8: Performance results of applying several reformulation approaches on TREC PM topics. Statistical significance (paired t-test) over the baseline is denoted by * for $p < 0.05$ and ** for $p < 0.01$. Bold-faced values represent the approaches with maximum performance.

baseline reaching up to 22.6% performance increase with P@5, 19.8 % with P@10, and 20.0% with nDCG@10 (statistically significant with for all three metrics), and to an upper bound performance up 47% over the baseline. It also resulted in a positive performance impact for up to 19 queries (out of 30) as shown in Figure 3.14(a). Moreover, even when considering applying only a single refinement task (i.e., weight reduction), our approach seems to be very useful and can result in a significant improvement achieving up to 17.7% performance increase (statistically significant for all metrics).

For applying concept expansion using the domain-specific ontology, our results are shown to be less consistent than applying query refinements. On the one hand, we see that using MeSH resource to expand diseases with their related terms resulted in a significant performance improvement achieving performance increase of 11.3% with P@5, 13.2% with P@10, and 11.5% with nDCG@10 over the baseline (statistically significant for P@10 and nDCG@10). Also, as shown in Figure 3.14(a), this approach resulted in a positive impact of at least 13 topics out of the 30 topics included in the dataset. On the other hand, taking the same approach, but expanding gene names with their different names from Genetics Home Reference resource is shown to have no effect on our topic set, but when tuning the reformulation parameters and selecting their best settings, we see an upper bound of this approach reaching up to 13.2% performance increase.

One reason to explain this effect is because of the fact that the provided gene symbols are more standardized than the disease names found in MeSH resource, and in fact, it is rare to find an abstract which discuss some aspects of a certain gene without mentioning the standardized symbol for that gene. In other words, we would expect a noticeable impact of our gene disambiguation only when the clinical cases contain some gene names other than the standard symbols. To summarize, looking at Figure 3.13 and Figure 3.14, we see a real impact of applying different sets of reformulation approaches to precision medicine queries. Similar to what we discussed for clinical decision support queries in Section 3.2.3.2, both query refinements and MeSH expansions seem to have



(a) Parameter selection with cross validation

(b) Parameter selection with best settings

Figure 3.14: Per query performance comparison with P@10 for each of the reformulation approaches and the baseline. Two cases are considered for selecting reformulation parameters.

more significant effect in promoting retrieval effectiveness for this category of search tasks.

3.4 Conclusions

In this chapter, we explored the effectiveness of applying a diverse set of query reformulations to promote search tasks for the clinical domain. Mainly, we presented two approaches, query refinements and ontology-based expansion, which can be used to replace manual query re-writing. In our experiments, using two variety of clinical search tasks, we showed that using those reformulations with some powerful tools such as semantic mapping can lead to a significant improvement in the performance of clinical based retrieval systems. Our work has the potential to be further improved by incorporating other useful features such as applying learning approach to estimate concept ratings instead of using the weighted voting. In fact, some of these potential improvements will be thoroughly discussed in the following chapter.

Chapter 4

FRAMEWORK FOR LEARNING TO PERFORM QUERY REFORMULATIONS

In the previous chapter, we evaluated the effectiveness of reformulating clinical queries by introducing a method for applying a set of reformulation tasks such as query re-writing and term expansion. Our method relies on the semantic mapping of the concepts in clinical queries to decide which reformulation task is more suitable to be applied in order to improve the quality of search results and to lead to retrieving more relevant articles. However, the main limitation of this method is that it involves some manual effort while assigning semantic types to the relevance bags as well as choosing the thresholds for making rating decisions. We observe that several other approaches, which attempt to perform reformulations on clinical queries, also share the same limitation by relying on either completely manual or semi-automatic methodologies [86, 88, 71, 120]. For instance, Limsopatham et al. [71] and Palotti et al. [88], introduced a semi-automatic approach which uses semantic mapping to identify different kinds of concepts in medical queries. Then, a set of modifications is applied based on the semantic types of the identified concepts. In both approaches, manual settings were used to reduce the full set of semantic types into a subset of types more related to the clinical domain.

Other work by Wan et al. [120] involved conducting a study to compare the effectiveness of both automatic and manual query reformulations. In that study, a domain expert, a clinician, was asked to manually modify a set of 30 queries by simply removing non-relevant terms and those terms deemed not necessary. The outcomes from the study suggest that search results can be substantially improved by performing manual query modification. One shortcoming of this study is that no user feedback modeling nor any generalization of how to transform manual models into an automatic one is discussed as a result of the conducted experiments.

As seen, none of the previous approaches (including our approach proposed in Chapter 3) attempted to transform user feedback, which can be provided as set of modifications that are applied to each query, into learned models and using these models as a part of recommending suitable reformulations to be applied while performing search for future information seeking tasks. In this chapter, aiming to tackle this limitation, we show our preliminary work for introducing a method which can be used to learn a model from simulated user feedback (provided as a rating to indicate the relevance of each concept), and use that model to estimate the rating of new unseen concepts in future search queries [5]. In our experiments, we show that by using simulated clinician feedback, we can learn several models that can estimate concept ratings with high accuracy. Furthermore, we integrate those models with our baseline search system and show how the query reformulations which are based on the estimated concept ratings can promote search results with high retrieval effectiveness.

4.1 Methodology

In this section, we introduce and discuss our proposed approach for learning to perform reformulations. Generally, our approach is applied by following three main steps. First, using semantic mapping tools such as MetaMap, we parse the terms and the phrases provided in the clinical case, and we annotate those terms by assigning each of them to a distinct set of predefined semantic types. Then, using either real user study or simulated user feedback, we transform the mapped terms from concept-mapping representation into the concept-feature representation to perform model learning and testing. Lastly, applying several machine learning algorithms, we can learn effective models that can be used to predict the ratings for new unseen concepts. Figure 4.1 illustrates applying our approach to learn/predict the ratings of concepts in a given query.

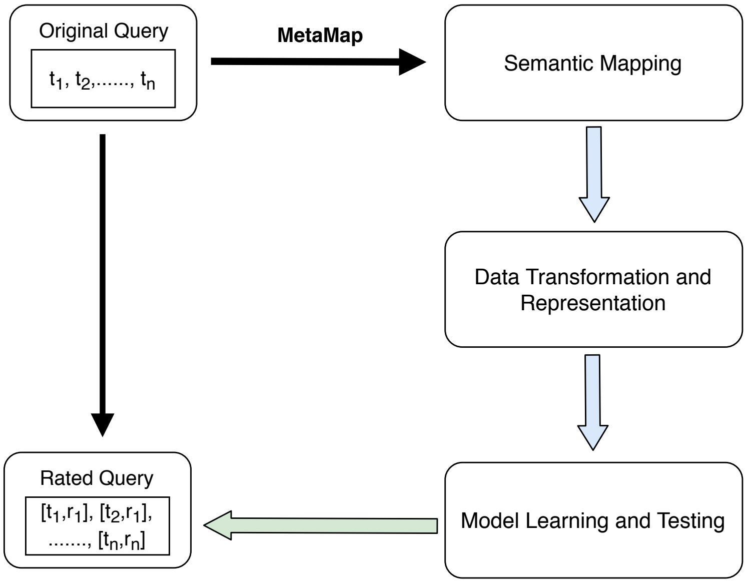

Original Query
t1, t2,......, tn
MetaMap
Semantic Mapping
Data Transformation and Representation
Model Learning and Testing
Rated Query
[t1,r1], [t2,r1], ..., [tn,rn]

Figure 4.1: The main steps to learn/predict the ratings of concepts in clinical queries.

4.1.1 Semantic Mapping of Clinical Concepts

Given a biomedical text, we aim to interpret this text such that the most informative parts are extracted and highlighted. More specifically, for clinical queries, our aim is to identify the most relevant information about a patient such as clinical findings, lab test, and previous treatments. This is accomplished by applying semantic mapping which refers to annotating each concept (i.e., term or phrase) and associating the annotated concepts to generic "semantic types" such as *behavior*, *clinical drug*, *organism*, etc.

For its availability and simplicity, we use MetaMap [12] which uses advanced NLP techniques to analyze biomedical text and perform semantic mapping. MetaMap relies on the Unified Medical Language System (UMLS) knowledge library to map clinical concepts to 133 distinct semantic types[1] such that each concept is mapped to one or more semantic types (out of those 133 predefined types). For example, consider the clinical query "*8-year-old boy with 2 days of loose stools, fever, and cough after returning from a trip to Colorado. Chest x-ray shows bilateral lung infiltrate*",

[1] https://metamap.nlm.nih.gov/Docs/SemanticTypes_2013AA.txt

Concept	Semantic Mapping
8-Year-Old	[Temporal Concept]
Boy	[Age Group]
2 Days	[Finding]|[Temporal Concept]
Loose Stools	[Sign or Symptom]|[Finding]
Fever	[Finding]
Cough	[Sign or Symptom]|[Finding]|[Pathologic Function]
Return Trip	[Qualitative Concept]|[Phenomenon or Process]
Colorado	[Geographic Area]
Chest X-ray	[Diagnostic Procedure]|[Clinical Attribute]
Show	[Intellectual Product]
Bilateral	[Spatial Concept]
Lung Infiltrate	[Finding]|[Disease or Syndrome]

Table 4.1: Using MetaMap to annotate clinical concepts and semantically map each concept to several semantic types.

MetaMap annotates the concepts of such query and then associates each concept with one or multiple UMLS semantic types. Table 4.1, illustrates the annotated concepts and the associated mapping of that query.

4.1.2 Learning From User Feedback

Although semantic mapping using MetaMap is useful in analyzing biomedical texts, one limitation is that it does not provide any additional information of which part of the text is more relevant to the task that a user is trying to accomplish. When performing clinical based retrieval tasks, providing some hints to retrieval systems about the most relevant findings of a clinical case is beneficial to focus on such findings and eliminate other findings that are not relevant. Eventually, this leads to improving search results.

We propose to overcome this limitation by leveraging domain experts' feedback about different concepts in clinical queries. The feedback, provided as ratings, for several concepts extracted from previous search queries is used to learn a model and then predict the ratings of new concepts in new (unseen) queries. Those ratings exemplify the importance of the concepts such that higher rating values indicate higher relevance whereas low ratings indicate lower or no relevance.

Our approach for learning user feedback works as follows. First, we assume that $Q_{train} = \begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_l \end{bmatrix}$ is the set of queries used for training and included in our user study; each query q_k is represented by a set of concepts such that $q_k = \begin{bmatrix} c_1 & c_2 & c_3 & \cdots & c_m \end{bmatrix}$. Also, each concept c_i in the query is represented by a vector of semantic types which it exemplifies $c_i = \begin{bmatrix} s_1 & s_2 & s_3 & \cdots & s_n \end{bmatrix}$ and each s_j can take either *True* or *False*, indicating whether a concept c_i is mapped to a semantic type s_j or not, respectively (also, instead of the Boolean values, we can use integer values to indicate the degree of confidence that a concept is mapped to certain semantic types). Note that n is fixed for all concepts and because we rely on UMLS knowledge base, $n = 133$ such that each s_j in c_i corresponds to a single semantic type in UMLS. In addition, we assume that each clinician in this study is shown the set of queries in Q_{train} and for each concept c_i in query q_k, he or she will evaluate the relevance of that concept to clinical decision making using 4 point scale such that (3) represents "definitely relevant", (2) represents "potentially relevant", (1) represents "potentially not relevant", whereas (0) represents "definitely not relevant".

In cases where there is a disagreement between the assessors (i.e., when clinicians provide distinct ratings for the same concept), we could either apply majority voting or average those values to select the rating for that concept. Finally, the ratings collected from this evaluation are matched with each concept in our data set. As a result, we represent each concept c_i by the vector $\begin{bmatrix} s_1 & s_2 & s_3 & \cdots & s_n & |y_i \end{bmatrix}$ where $y_i \in (0{,}1{,}2{,}3)$ and $s_j \in$ (*True, False*). The resultant data set for all concepts of the queries in Q_{train} will be:

$$\begin{Bmatrix} s_{11} & s_{12} & s_{13} & \dots & s_{1n} & |y_1 \\ s_{21} & s_{22} & s_{23} & \dots & s_{2n} & |y_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ s_{i1} & s_{i2} & s_{i3} & \dots & s_{in} & |y_i \end{Bmatrix}$$

where each row represents a single concept and its provided rating.

Our problem can be considered as a classification problem where we train our classifier with several concepts using the 133 Boolean values for semantic types as features, and we use the provided user rating as a target class. For new queries that are not part of the training, we predict the rating of all concepts in those queries using the mapped semantic types and by applying our learned classifier. We consider using three classification algorithms: multinominal logistic regression (Logistic), support vector machines (SVM), and deep learned neural networks (DeepANN). For logistic and SVM, we use *Weka* [48], and for DeepANN, we use *deepnet*[2] package in R.

4.1.3 Simulating Clinician Feedback

With the absence of actual/real user feedback, we choose to simulate clinician ratings in order to train our model (conducting a user study is left for proposed future work). We assume that a concept c in a clinical query will be rated highly by a clinician if it has been mapped to semantic types that are more related to clinical diagnosis procedure.

We apply a weighted voting approach that is discussed in Section 3.2.2.1 which rewards concepts that are clinically-related and penalizes those which are not. We manually place each of the 133 semantic types obtained from MetaMap into one of three bags: one for the semantic types we have identified as clinically-related (e.g., clinical findings, symptoms, and diseases), one for semantic types with lower clinical relation (e.g. human daily or recreational activities), and one for types with very weak relation to the clinical domain (e.g., temporal and geographical types). Our simulation works by applying MetaMap to a clinical query such as the one presented in Table 4.1, and for each extracted concept we output a list of possible semantic types. The presence of those types in the three bags described above is then used to compute a "voting sum" for the concept by adding the corresponding bag weight for each semantic type that appears in that bag. A concept with semantic types that appear mostly in the clinically-related bag will have higher weight than a concept with semantic types that

[2] https://cran.r-project.org/web/packages/deepnet/index.html

are split evenly between clinically-related and weakly clinical; the latter will have higher weight than a concept that maps to semantic types mostly in the weakly clinical bag. We compute the voting sum for each concept c according to the following formula:

$$Sum(c) = \sum_{i=1}^{K} W_i C_i \tag{4.1}$$

where K represents the number of bags, and W_i and C_i represent the weight and the number of mapped semantic types in bag i. We use the result of this voting to decide what rating should be given to concept c based on three thresholds (t_1, t_2, and t_3) as follows (see Chapter 3 for more details).

$$Rate(c) = \begin{cases} 3, & \text{if } Sum(c) \geq t_3 \\ 2, & \text{if } t_3 > Sum(c) \geq t_2 \\ 1, & \text{if } t_2 > Sum(c) > t_1 \\ 0, & \text{if } Sum(c) \leq t_1 \end{cases} \tag{4.2}$$

4.2 Evaluation

In this section, we present our evaluation of the proposed approach. First, we describe the datasets used in our evaluation. Then, we discuss the experimental designs and settings. Finally, we summarize our results and provide some discussion.

4.2.1 Test Data

We use the concepts extracted from the topic sets for the Text Retrieval Conference (TREC) Clinical Decision Support (CDS) track [104]. Initially, each topic represents a short clinical summary of a medical case and consists of several clinical concepts describing specific aspects of patient health such as symptoms, health conditions, lab test, and previous treatments. We use the three topic sets for the years 2014, 2015, and 2016 of the track (30 topics for each set) such that we extracted the concepts in those topics using MetaMap as described in Section 4.1.1. This resulted in a total 421 concepts for 2014, 361 concepts for 2015, and 538 concepts for 2016. We then apply our simulated approach to each concept and extract semantic types as

	Simulated Ratings			
Topic set	**'0'**	**'1'**	**'2'**	**'3'**
CDS 2014	14.44%	8.96%	61.43%	15.17%
CDS 2015	12.73%	7.39%	60.78%	19.09%
CDS 2016	10.84%	8.73%	59.79%	20.63%

Table 4.2: The proportion of clinical concepts under each rating category for 2014, 2015, 2016 TREC CDS track topic sets

well as their recommended ratings. Later, those ratings in addition to the transformed semantic types will be used as instances for training several models. Table 4.2 shows the proportion of the concepts under each rating category for the three data sets.

4.2.2 Experimental Settings

After generating the simulated feedback as discussed in Section 4.1.3, we train several models using three classifiers logistic, SVM, and DeepANN with default settings for both *Weka* and *deepnet*. Because deep learning requires specifying the structure of the neural network, we construct a network with one hidden layer containing 70 units while relying on prediction error to make this selection. For each topic set, we use six-fold cross-validation to train each classifier such that in each fold, we train with the concepts of 25 topics and test with the concepts of 5 topics.

We evaluate our classifiers using four metrics: accuracy, precision, recall, and AUC. Accuracy measures the proportion of correctly classified instances to the overall number of instances in the data. Precision measures the proportion of correctly classified instances for one class to the overall instances labeled by a classifier as the same class. Recall measures the proportion of correctly classified instances for one class to all instances for that class in the data. Lastly, AUC measures the area under the curve for a single point on the Operator Characteristic Curve (ROC). Because we have a multi-class problem, we use macro-averaging for computing precision and recall [105]; also we use one variation of averaging discussed in [49] to compute AUC. The equations for precision, recall, and AUC for multi-class problems are defined as follows.

$$\text{Precision} = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}}{l} \tag{4.3}$$

$$\text{Recall} = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l} \tag{4.4}$$

$$\text{AUC} = \frac{\sum_{i=1}^{l} \frac{1}{2}\left(\frac{tp_i}{tp_i + fn_i} + \frac{tn_i}{tn_i + fp_i}\right)}{l} \tag{4.5}$$

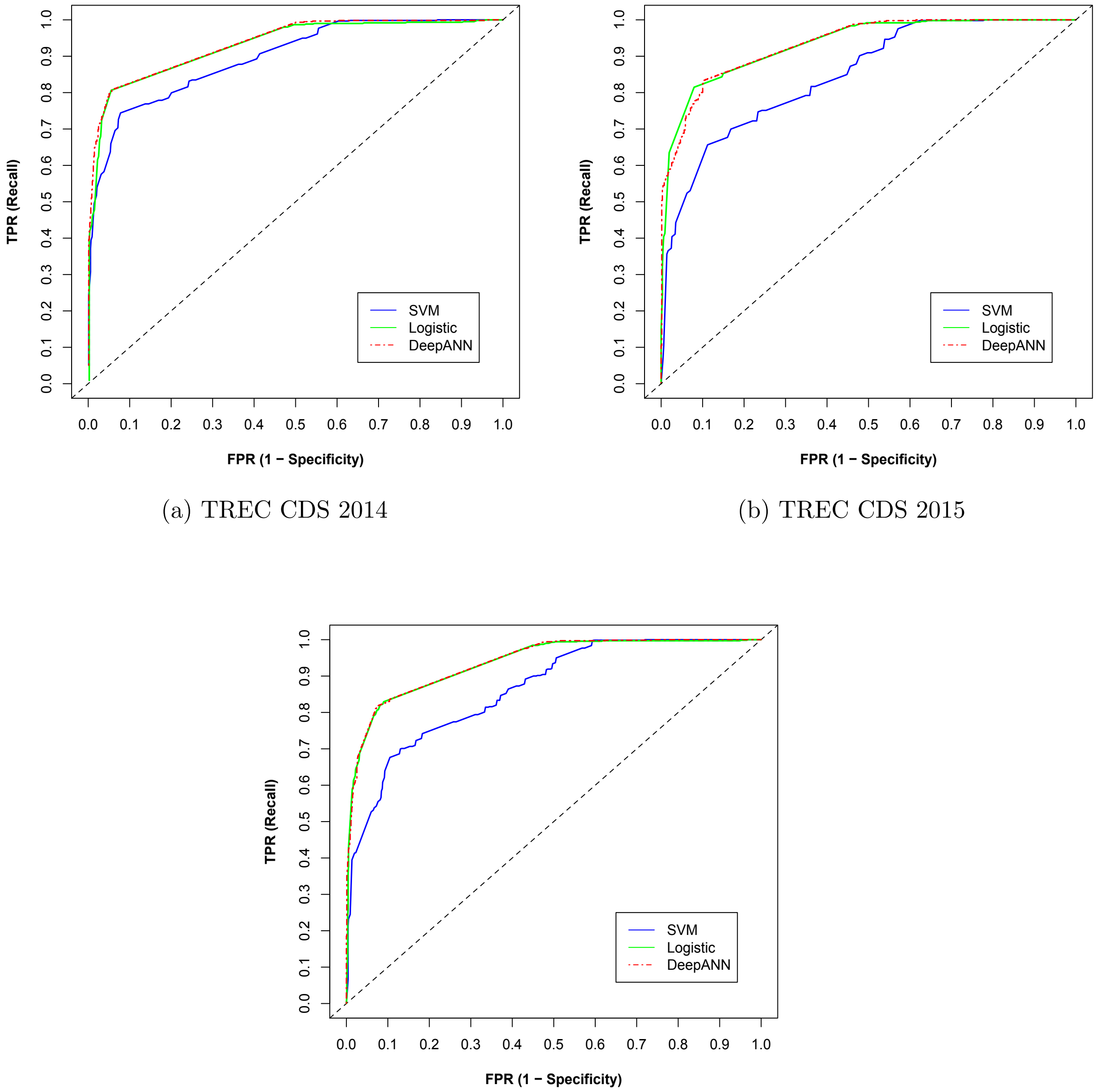
where l represents the number of classes in the training set, tp is the number of true positive instances, fp is the number of false positive instances, tn is the number of true negative instances, and fn is the number of false negatives.

4.2.3 Results and Analysis

Table 4.3 presents our results for training three classifiers on TREC CDS topic sets, and Table 4.4 shows the results of statistical significance among classifiers using a pairwise one-sided t-test. Also, Figure 4.2 compares the effectiveness of the three classifiers in making correct predictions using Operator Characteristic Curve (ROC), which plots true positive rate (recall) as a function of false positive rate.

From Table 4.3 and Table 4.4, we see that in terms of accuracy, all the three classifiers seem to perform well in minimizing classification error and are shown to achieve high accuracy levels that are above 90% (DeepANN is shown to be slightly higher than both Logistic and SVM, and it is statistically significant over SVM only). For precision, the case is somewhat different as shown that both SVM and Logistic achieved comparable performance levels (mostly, they are below higher than 80% but less than 90%) whereas DeepANN is shown to outperform the other classifier with a precision level that is above 90% (statistically significant over Logistic regression only).

On the other hand, for “recall” and “AUC” metrics, we see a major change in the performance between the three classifiers. In particular, our SVM classifier is shown



(a) TREC CDS 2014

(b) TREC CDS 2015

(c) TREC CDS 2016

Figure 4.2: Operator Characteristic Curve (ROC), comparing the true positive rate to the false positive rate, is plotted for the three classifiers SVM, Logistic, and DeepANN (averaged among the four classes in the training set).

to perform extremely poor compared to DeepANN and Logistic. Also, the difference between both DeepANN and Logistic (on one hand) and between SVM is shown to be

Topic set	Classifier	Accuracy	Precision	Recall	AUC
CDS 2014	SVM	93.78%	**96.10%**	70.50%	76.29%
	Logistic	94.25%	86.35%	93.85%	86.21%
	DeepANN	**98.09%**	94.21%	**98.77%**	**92.15%**
CDS 2015	SVM	91.43%	84.91%	58.51%	70.12%
	Logistic	93.71%	83.00%	**96.28%**	**84.65%**
	DeepANN	**95.43%**	**92.20%**	92.76%	82.27%
CDS 2016	SVM	91.86%	88.87%	69.02%	73.20%
	Logistic	94.17%	83.73%	95.81%	84.81%
	DeepANN	**98.84%**	**98.58%**	**95.91%**	**89.01%**

Table 4.3: The classification accuracy results for three classifiers SVM, Logistic, and DeepANN using concepts ratings from three TREC CDS topic sets. Boldfaced values indicate the classifier with best performance for that metric.

Classifier	Accuracy	Precision	Recall	AUC
SVM	—	—	—	—
Logistic	—	—	>SVM†	>SVM††
DeepANN	>SVM†	>Logistic†	>SVM††	>SVM††

Table 4.4: Statistical significance results using pairwise one-sided t-test. >classifier† and >classifier†† denote statistical significance over "classifier" with $p < 0.05$ and $p < 0.01$, respectively.

statistically significant with $p < 0.01$ on both metrics for DeepANN, whereas $p < 0.05$ and $p < 0.01$ on recall and AUC, respectively for Logistic Regression. Moreover, from this result and by looking at the comparison between the three classifiers in Figure 4.2, we can conclude that DeepANN and Logistic classifiers are better than SVM in capturing more accurate ratings and both classifiers seem to be comparable even though in few instances DeepANN is shown to be slightly better.

4.3 Application: From User Models to Clinical based Retrieval

In this section, we discuss how to integrate our proposed approach described in Section 4.1 with a clinical search system and use the predicted ratings to apply suitable reformulation tasks to each given clinical query. In Section 4.3.1, we describe our experimental settings and in Section 4.3.2, we present and analyze our results

4.3.1 Experimental Settings

As discussed in Section 2.1.1, information retrieval systems rank the retrieved documents according to their relevance to a given user query using a scoring function that is computed for each document. Integrating our learned models with retrieval systems is performed by modifying such scoring function. In our experiments, we use Terrier IR platform [85] and rely on the $InExpB2$ DFR based ranking model [8] which initially ranks the documents according to the following scoring function:

$$Score(d, Q) = \sum_{t \in Q} qtw \cdot w(t, d) \tag{4.6}$$

where t is a term in a query Q, qtw is the term weight in Q, and $w(t, d)$ is the term weight in a document d computed as described in Section 2.1.1.

To perform our experiment, first, we follow the steps discussed in Section 3.2.3.1 to build the index and the baseline system used to perform clinical search tasks. Then, we learn three classification models SVM. Logistic regression, and DeepANN using the topic sets for TREC CDS 2014, 2015, 2016. For each query in our data set, we extract the concepts of that query, and we use the ratings produced by the learned model to recommend the suitable refinement task that will be applied to the extracted concepts. For instance, if a concept in a query receives rating $r = 0$ (i.e., considered to be not relevant), then we remove that concept from that original query. If a concept gets $r = 1$ (i.e., may or may not be relevant), we keep it; however, the weight of that concept will be reduced. Also, if the concept's rating $r = 2$ (i.e., relevant), then we keep that concept with no change. Lastly, if it receives rating $r = 3$ (i.e., definitely relevant concept), then we increase its weight; also, we boost the score of all documents which has that concept in the introductory parts (e.g., abstract) because such documents are most likely to be relevant in answering that query. All of the actions above can be captured by modifying the initial scoring function 4.6 as follows:

$$Score(d, Q) = (1 - \lambda) * \Big[\sum_{t \in Q_\star} qtw \cdot w(t, d) + \sum_{t \in Q_b} w_b \cdot w(t, d) - \sum_{t \in Q_r} w_r \cdot w(t, d) \Big] + \lambda \Big[\sum_{t \in Q_b} w_b \cdot w(t, d) \Big] \quad (4.7)$$

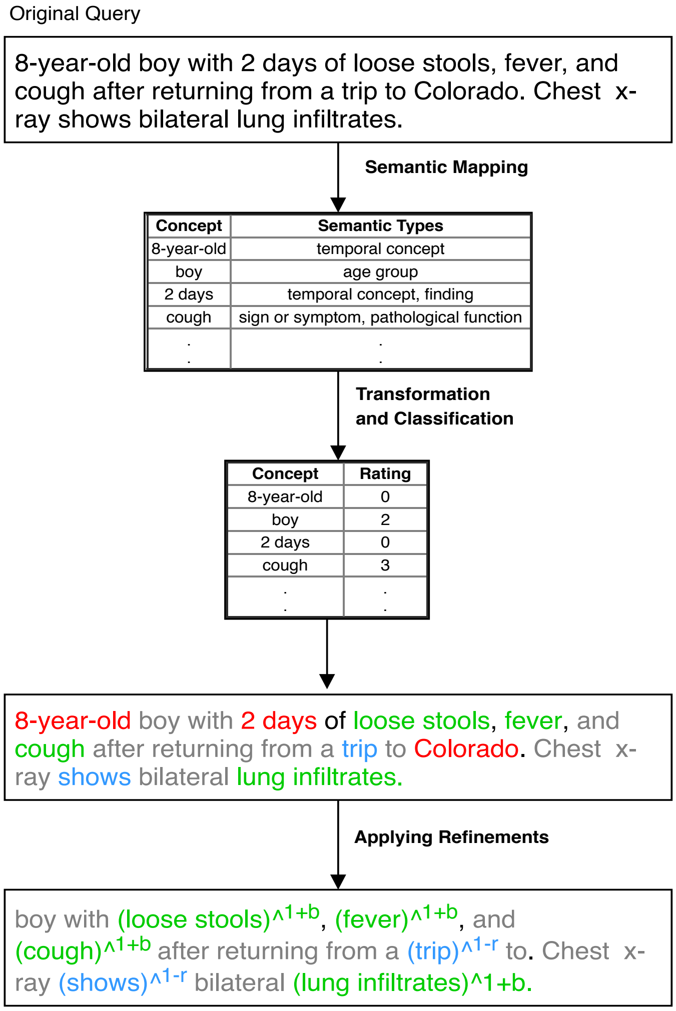
where $Q_\star$ represents all the query terms after removing concepts rated 0, Q_b represents the set of terms for the concepts to be boosted and w_b is the boosting weight. Also, Q_r represents the set of terms for the reduced concepts and w_r is the reduction weight. Finally, λ is used in the equation above to weight the abstract field in document d.

The last step we perform after applying each of the refinements above is submitting the given query to our clinical search system and ranking the retrieved documents using equation 4.7. Figure 4.3 summarizes the steps we perform to extract query concepts and apply reformulations based on learned models.

4.3.2 Results and Analysis

Table 4.5 summarizes the retrieval performance results after integrating the learned user models with our retrieval system. We use the ratings predicted by the three classifiers SVM, Logistic, and DeepANN for the concepts in TREC CDS 2014, 2015, and 2016 topic sets. We evaluate our results using precision at rank 10 (P@10) and normalized discounted cumulative gain (nDCG@10) [58]. Also, we compare our results to a baseline which does not apply query reformulations. Lastly, we set the parameters in equation 3.6 using both best settings (upper bound performance estimation) and 6-fold cross validation with P@10 metric to select the parameter values which give the highest average performance.

Form Table 4.5, we see that integrating our approach, which transforms manual reformulation methods into learned models, indeed led to significantly increasing the retrieval effectiveness of our baseline system, reaching up to 32.6% performance increase for SVM, 42.2% for Logistic, and up to 44.6% for DeepANN. Also, since we used the manual approach used in Chapter 3 to generate the simulated ratings, we see that the results our model learning correlate very well with the results of the previous


Original Query
8-year-old boy with 2 days of loose stools, fever, and cough after returning from a trip to Colorado. Chest x-ray shows bilateral lung infiltrates.
Semantic Mapping
Transformation and Classification
8-year-old boy with 2 days of loose stools, fever, and cough after returning from a trip to Colorado. Chest x-ray shows bilateral lung infiltrates.
Applying Refinements
boy with (loose stools)^1+b, (fever)^1+b, and (cough)^1+b after returning from a (trip)^1-r to. Chest x-ray (shows)^1-r bilateral (lung infiltrates)^1+b.

Concept	Semantic Types
8-year-old	temporal concept
boy	age group
2 days	temporal concept, finding
cough	sign or symptom, pathological function
. .	. .

Concept	Rating
8-year-old	0
boy	2
2 days	0
cough	3
. .	. .

Figure 4.3: The complete process for applying a set of refinement tasks to a clinical query by using the ratings produced by learned models. Annotation with different colors is used to distinguish different ratings.

experiments discussed in Section 3.2.3.2. This suggests that the proposed approach can be effective in capturing and learning some properties from manually designed models.

On the other hand, we observe that the retrieval effectiveness of a search system is sensitive to which learning algorithm is used to learn concept ratings. In particular, looking at the results in Table 4.3, we see that SVM classifier, which has the lowest

CDS 2014				
System	**P@10 (CV)**	**P@10 (B)**	**nDCG@10 (CV)**	**nDCG@10 (B)**
Baseline	0.3500	0.3500	0.2899	0.2899
SVM	0.4000	0.4533	0.3158	0.3556
Logistic	0.4267	0.4700	**0.3340**	0.3771
DeepANN	**0.4300**	**0.4900**	0.3337	**0.3887**
CDS 2015				
System	**P@10 (CV)**	**P@10 (B)**	**nDCG@10 (CV)**	**nDCG@10 (B)**
Baseline	**0.4100**	0.4100	0.2706	0.2706
SVM	0.4067	0.4533	0.2731	0.2976
Logistic	0.3967	0.4533	**0.2762**	0.3015
DeepANN	0.3933	**0.4733**	0.2755	**0.3124**
CDS 2016				
System	**P@10 (CV)**	**P@10 (B)**	**nDCG@10 (CV)**	**nDCG@10 (B)**
Baseline	0.2766	0.2766	0.1964	0.1964
SVM	0.2733	0.3667	0.1923	0.2360
Logistic	0.3033	0.3933	0.1958	0.2468
DeepANN	**0.3133**	**0.4000**	**0.2081**	**0.2517**

Table 4.5: The retrieval effectiveness results for applying ratings to clinical topics using several classification methods. “B” and “CV” indicate whether best settings or cross validation, respectively is used for parameter settings. Bold-faced values indicate the system with the highest performance.

classification accuracy across all the topic sets as shown previously in Table 4.3, also resulted in the lowest retrieval effectiveness compared to DeepANN and Logistic for almost all the topic sets. Moreover, in some cases, the difference between SVM and the two other classifiers can be very significant as shown for 2014 and 2016 sets (the difference between SVM and DeepANN reaches to 14% with P@10 for 2016). From this analysis, we suggest that the failure/success of those models in leveraging search results will be tied to whether the used classification algorithm fits training data very well and to whether or not the data set is generated in a consistent way.

4.4 Conclusion

As discussed throughout this chapter, one of the main limitations of the current reformulation techniques in the clinical search is that they rely on intensive manual settings in order to work effectively. In this chapter, we attempt to reduce and eliminate this limitation by proposing an approach for learning to perform reformulations based on simulated user feedback. Our method is shown to be effective in predicting user ratings and in improving the effectiveness of clinical text retrieval. Generally, our work has the potential to be enhanced by including a user study and learning models using actual user feedback. Also, we plan to extend our approach to other clinical text understanding tasks such as text summarization and concept annotation.

Chapter 5

THE USEFULNESS OF LEARNING TO RANK FOR CLINICAL SEARCH TASKS

In Chapter 2, we discussed some previous work for applying Learning to Rank (LtR) in a variety of search tasks. In this chapter, we explore the usefulness of applying learning to rank in clinical search tasks as it has shown to be very effective in leveraging the performance of search tasks in several domains including those for web search [76, 41, 80]. The major advantage of using LtR is having the ability to incorporate a large number of features (strong and weak) into ranking models and using these models to produce search results that are expected to be better than what is produced by traditional models (e.g., BM25 and LM) that only compute a match between document and query [72].

In this chapter, we present a general approach for applying a range of LtR techniques to clinical search tasks [3]. Our work makes the following contributions: first, it introduces an ensemble of generic and domain-specific features which can be used to promote LtR in the search tasks for clinical decision making, and second, it proposes a greedy feature selection method which can help in limiting the effect of model overfitting when too many features are incorporated in model training. Moreover, by examining our approach with several LtR algorithms, we see that indeed, this framework can increase the effectiveness of LtR learners. We apply our approach to a sample of clinical queries and show that it can significantly promote search results, leading to a performance increase of 28% over a baseline clinical search system.

5.1 Methodology

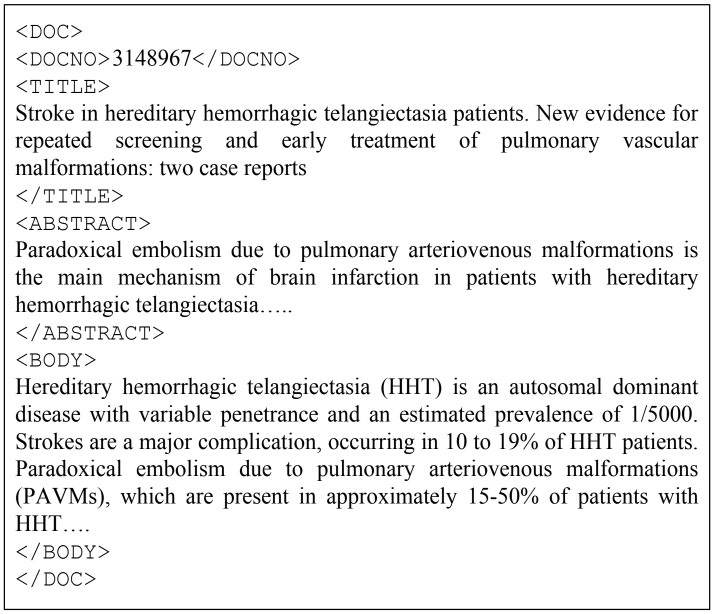
Before proceeding further to describe the main components of our methodology, we explain our ranking problem. Performing LtR based ranking requires learning a model using a sample of ground-truth values that represent our training instances. Those instances consist of $\mathcal{Q} = \begin{bmatrix} q_1, & q_2, & q_3, \cdots \end{bmatrix}$ as a set of queries (e.g., medical cases) where each query $q_i \in \mathcal{Q}$ is associated with $\mathcal{D} = \begin{bmatrix} d_1, & d_2, & d_3, \cdots \end{bmatrix}$ which is a set of humanly judged documents (in our case, biomedical articles). Therefore, each query-document pair (q_i, d_j) will be labeled with a relevance judgment y_i, an integer with a fixed range (e.g., $y_i \in [0, 1, 2]$), indicating the relevance level of document d_j to query q_i. By applying LtR learning algorithms on those labels, we can learn effective ranking models relying on a large number of features extracted from query-document pairs.

Next, we describe the data and the feature set we use in our model learning. Also, we describe the learning algorithms used in our study. Later in this section, we present our method for performing feature selection.

5.1.1 Dataset

We use the collection and topic sets for the TREC CDS track [93]. The collection consists of over 1.25 million biomedical articles that are sampled from PubMed Central (PMC). We use all 90 topics from three years of the track (2014-2016, 30 topics per year). Each topic includes a free-text query and summarizes various aspects of a patient's medical history which includes findings, lab tests, and previous treatments (e.g., "*58-year-old female non-smoker with left lung mass on x-ray. Head CT shows a solitary right frontal lobe mass*").

We process each article in the collection by extracting the most informative textual content and create a corresponding ready-to-index document. Each processed document consists of the four fields: `<DOCNO>, <TITLE>, <ABSTRACT>,` and `<BODY>`. `<DOCNO>` represents the article's PMID in PMC library, `<TITLE>` and `<ABSTRACT>` represent the title and the abstract of the article respectively, while `<BODY>` represents all



```
<DOC>
<DOCNO>3148967</DOCNO>
<TITLE>
Stroke in hereditary hemorrhagic telangiectasia patients. New evidence for
repeated screening and early treatment of pulmonary vascular
malformations: two case reports
</TITLE>
<ABSTRACT>
Paradoxical embolism due to pulmonary arteriovenous malformations is
the main mechanism of brain infarction in patients with hereditary
hemorrhagic telangiectasia…..
</ABSTRACT>
<BODY>
Hereditary hemorrhagic telangiectasia (HHT) is an autosomal dominant
disease with variable penetrance and an estimated prevalence of 1/5000.
Strokes are a major complication, occurring in 10 to 19% of HHT patients.
Paradoxical embolism due to pulmonary arteriovenous malformations
(PAVMs), which are present in approximately 15-50% of patients with
HHT….
</BODY>
</DOC>
```

Figure 5.1: An example of a processed biomedical article which contains the most informative fields.

other textual content which is not included in the previous fields. Figure 5.1 illustrates a sample biomedical article after extracting the four fields. Lastly, we index all the documents using Terrier IR platform [73] while applying Porter stemming algorithm and default stopword list.

5.1.2 Features

We rely on a variety of features to train clinical based LtR learners. We group our features into three categories: general IR features used in many search domains, features of the query, and clinical concept features more specific to the biomedical domain.

5.1.2.1 General IR Statistics

We use a set of standard features that are applied in web and other search tasks. Specifically, we consider the features used in the LETOR dataset [91] since they have been shown to be effective in many LtR applications as suggested by Qin et al.

Feature	Description	Source	Total
TFIDF	Vector space ranking model based on TFIDF weighting computed as defined in Section 2.1.1.2	documents and queries	4
BM25	Okapi BM25 ranking model computed as defined in Section 2.1.1.3	documents and queries	4
LM	Language ranking model computed as defined in Section 2.1.1.4	documents and queries	4
TF	Number of occurrences of query terms within a document or a field	documents and queries	4
IDF	Inverse document/field frequency of query terms	documents and queries	4
TermCount	Number of terms in a document or a field	documents	4

Table 5.1: A set of 24 features representing general IR statistics that are used in LETOR dataset.

[91], and Liu et al. [72]. The feature set for this category includes retrieval model statistics (e.g., BM25 and LM), query and document term matching statistics (e.g., term frequency and IDF), and other essential statistics such as document and field lengths. We compute each of these features for the whole document and the three fields <TITLE>, <ABSTRACT>, and <BODY> (i.e., four variations are computed for each feature). Moreover, we exclude from our set any feature that is not applicable to our dataset (e.g., click features since we do not have any user click information). This results in a total of 24 features under this category which are summarized in Table 5.1.

5.1.2.2 Query Features

We use 11 types of query features in our learning to capture some performance properties of each query. The features we use for this category are query performance predictors used in IR literature to evaluate different aspects of user queries. For example, some features may evaluate the relatedness between query terms (e.g., AvgPMI [50] and NSCQ [133]), while other features may measure the level of generality of a given query (e.g., clarity and scope [51]). In order to increase the effectiveness of these features, we multiply each one by each of the three document retrieval scores (BM25, LM, and TFIDF), producing three distinct values of the feature for each query/document pair. In the total, this results in 33 features that are presented in Table 5.2.

5.1.2.3 Clinical Concepts

The final set contains some features more specific to biomedical and clinical search purposes. We use MetaMap [12], a biomedical text analysis tool, to extract clinical concepts (i.e., terms and phrases) from both queries and documents such that only concepts occurring in both query and document are counted. Furthermore, because MetaMap relies on the Unified Medical Language System (UMLS)[1] in identifying biomedical concepts, we consider only the concepts which belong to 16 semantic types (e.g., "[Finding], [Sign or Symptom], and [Disease or Syndrome]") as they have shown to be more relevant to clinical search tasks as suggested by Limsopatham et al. [71]. Lastly, we gather the statistics from the concept extraction process (i.e., count the number of matched concepts under each semantic type for each query/document pair) and use them to generate 17 distinct features as shown in Table 5.4.

5.1.3 Learning Algorithms

Having presented the set of features that we consider in this work, we now provide a brief description of the algorithms used to learn those features. LtR algorithms can be differentiated according to their loss functions into pointwise, pairwise, and listwise (see [72] for a detailed overview). In this work, we try to experiment with several algorithms which have different loss functions, so we choose the following algorithms: random forests [16] (pointwise), RankBoost [41], LambdaMart [126] (pairwise), AdaRank [127], and Coordinate Ascent [80] (listwise). All of these algorithms are implemented in RankLib software package[2].

[1] https://www.nlm.nih.gov/research/umls

[2] https://sourceforge.net/p/lemur/wiki/RankLib

Feature	Equation	Source	Total
QL [51]	Number of terms in a query	queries	3
AvgPMI [50]	$\frac{\sum_{t_a,t_b\in Q}\log\left(\frac{P(t_a,t_b \mid C)}{P(t_a \mid C)*P(t_b \mid C)}\right)}{QL}$	queries	3
AvgICTF [51]	$\frac{\log\prod_{t\in Q}\frac{\lvert C\rvert}{ctf_t}}{QL}$	queries	3
AvgIDF [50]	$\frac{\sum_{t\in Q}\left(\frac{\log(N+0.5)}{\log(N+1)*df_t}\right)}{QL}$	queries	3
AvgNGD [27]	$\frac{\sum_{t_a,t_b\in Q}\left(\frac{Max(\log P(t_a),\log P(t_b))-\log P(t_a,t_b)}{\log(N*ADL)-Min(\log P(t_a),\log P(t_b))}\right)}{QL}$	queries	3
Clarity [51]	$\sum_{t\in Q}\frac{tf_t}{QL}*\log\left(\frac{tf_t*\lvert C\rvert}{ctf_t*QL}\right)$	queries	3
Scope [51]	$-\log\left(\frac{n_Q}{N}\right)$	queries	3
NSCQ [133]	$\frac{\sum_{t\in Q}(1+\log(ctf_t))*IDF(t)}{QL}$	queries	3
Sigma2 [133]	$\sum_{t\in Q}\sqrt{\frac{1}{df_t}\sum_{d\in D_t}(TFIDF(t,d)-w_t)^2}$	queries	3
Gamma2 [51]	$\frac{IDF_{max}}{IDF_{min}}$	queries	3
QC1 [52]	$\frac{1}{QL}\sum_{t\in Q}\left(\frac{\sum_{i\neq j\in 1..N}Sim(d_i,d_j)}{N(N-1)}\right)$	queries	3

Table 5.2: Query performance predictors used as features to capture distinct properties about queries.

t	is a term in query Q
tf_t	is the frequency of a term t in query Q
ctf_t	is the collection frequency of a term t
df_t	is the number of documents that contain term t
n_Q	is the number of documents that contain at least one query term
ADL	is the average document length
$\|C\|$	is the number of tokens in the whole collection
N	is the number of documents in the whole collection
$Cosine(d_i, d_j)$	is the cosine similarity between a pair of two documents
$P(t_a\|C)$	is the probability of term t_a occurring in collection C
$P(t_a, t_b\|C)$	is the probability of terms t_a and t_b occurring together in collection C
IDF_{max}	is the maximum inverse document frequency among query terms
IDF_{min}	is the minimum inverse document frequency among query terms

Table 5.3: Definitions for symbols in the equations of Table 5.2.

Feature	Description	Source	Total
Concept Types [71]	Total Number of matched UMLS concepts for a subset of 16 semantic types: [Body Location or Region], [Body Part, Organ, or Organ Component], [Clinical Drug], [Diagnostic Procedure], [Disease or Syndrome], [Finding], [Health Care Activity], [Injury or Poisoning], [Intellectual Product], [Medical Device], [Mental or Behavioral Dysfunction], [Neoplastic Process], [Pathologic Function], [Pharmacologic Substance], [Sign or Symptom], and [Therapeutic or Preventive Procedure]	documents and queries	16
Concepts Count	Total number of matched UMLS concepts between each document and a query	documents and queries	1

Table 5.4: Domain specific features representing the number of matched concepts under a subset of semantic types.

5.1.4 Feature Selection and Performance Optimization

Let $\mathcal{F} = \Big[f_1, f_2, f_3, \cdots, f_N\Big]$ is a set of features that are considered for learning a ranking model. Also, Let $i(f_i)$ is an importance measure (i.e., relevance) of feature f_i and $s(f_i, f_j)$ is a similarity measure between two features f_i and f_j in $\mathcal{F}$. In the next part, we introduce our method which relies on feature importance and similarity measures above to perform Feature Selection (FS).

WGAS. The procedure of Wrapper Greedy Search Algorithm (WGAS) is described in Algorithm1. WGAS starts by selecting an initial seed of k features which have the highest importance values and adding those features to the result set S. Then, a single LtR model is trained using the features in S after removing them from $\mathcal{F}$. After that, the algorithm enters a loop which is executed for $(n-k)$ times and in each iteration, it selects a feature f_x that maximize importance and minimize similarity, and removes it from $\mathcal{F}$; however, f_x may not be added to the final set S unless it shows a positive contribution to the performance of the learned model (i.e., by using validation data). Moreover, whenever a new feature is added to the set S, a penalization is conducted to the importance scores of all other features in F according to their similarities with the added feature. Similar to to Geng et al. [42], a free-parameter c is used to balance feature penalization. Ultimately, the algorithm stops as the number of examined features reaches to n.

5.1.4.1 Feature Importance and Similarity Measures

We need to define both the feature importance measure $i(f_i)$, and the symmetric similarity measure between two features $s(f_i, f_j)$ in our FS algorithm. For $i(f_i)$, we aim to find a function that estimates the relevance of feature f_i given the relevance labels of query/document pairs in our training data. Likewise, for $s(f_i, f_j)$, our goal is to estimate a robust correlation between features f_i and f_j.

Similar to Gigli et al. [44] and Geng et al. [42], we measure the importance of feature f_i by computing the average nDCG@10 [58] achieved by learning a model using that single feature (we learn a single LtR model for each feature f_i, and we evaluate that model using the average nDCG@10 on testing data such that the higher the value of nDCG@10, the higher the relevance of feature f_i). In addition, we measure feature similarity $s(f_i, f_j)$ by computing Kendall's correlation coefficients between the lists of the two features' query/document pair values. The higher the correlation between the two features the more similar they are.

Algorithm1: Wrapper GAS Algorithm (WGAS)

```
WGAS($\mathcal{F}$,$n$, $k$, $c$):
for $i = 1 : k$ do
    $f^{+} \leftarrow \operatorname*{argmax}_{f_x \in \mathcal{F}} i(f_x)$
    $S \leftarrow S + f^{+}$.
    $F \leftarrow F - f^{+}$.
end for
$L_{max} \leftarrow$ train model using features in $S$
for $i = 1 : n - k$ do
    $f^{+} \leftarrow \operatorname*{argmax}_{f_x \in \mathcal{F}} i(f_x)$
    $L_{curr} \leftarrow$ train model using features in $S + f^{+}$
    $F \leftarrow F - f^{+}$
    if $L_{curr} > L_{max}$ then
        $L_{max} \leftarrow L_{curr}$
        $S \leftarrow S + f^{+}$
        for each pair $(f^{+}, f_y) \in \mathcal{F}$ do
            $i(f_y) \leftarrow i(f^{+}) - c * s(f^{+}, f_y)$
        end for
    end if
end for
RETURN $S$
```

5.2 Evaluation

We now describe the experiments we perform to evaluate our approach for applying LtR ranking models to medical search tasks. First, we discuss the settings we apply to perform our evaluation. Then, in Section 5.2.2, we present and briefly summarize the results of our experiments. In Section 5.2.3 and Section 5.2.4, we conduct further evaluation of feature importance and feature selection, respectively. Finally, in Section 5.2.5, we provide additional statistical analysis and include further discussion of our results.

5.2.1 Experimental Setup

We use the data set described in Section 5.1.1 which consists of 1.25M articles and 90 clinical queries to evaluate our approach. We use the relevance judgments provided by TREC assessors as ground-truth labels for query/document pairs in our training data. Those labels represent the ratings given by the assessors to show the relevance of a document to a given query such that 0, which is the lowest rating, represents "definitely not relevant document" whereas 2, the highest, is representing "definitely relevant document". We index the target collection using Terrier IR [73] platform, and similar to Macdonald et al. [76], we use the resulting index to generate a sample for our training by using the BM25 model [97] (with default Terrier's settings) to retrieve the top 1,000 documents for each query. We compute each of the proposed features as described in Section 5.1.2 using the statistics of the resulting index. Ultimately, each query/document pair is represented by a vector of 74 features and a ground-truth label showing the relevance of that document to the given query.

We train several LtR models (i.e., random forests, RankBoost, LambdaMart, AdaRank, and Coordinate Ascent) using nine-fold cross validation while using a separate validation set for model selection and optimization: in each fold, the documents of 10 queries are used for testing whereas the documents of the remaining 80 queries are split between training and validation. Applying cross validation this way is standard for LtR learning which allows us to avoid any bias towards the documents of testing queries and to make sure that we are not partially learning from the testing data. We use RankLib package and we apply default package settings in our learning. Lastly, due to the effect of optimization metrics on the overall performance of some LtR learners (e.g., LambdaMART, AdaRank, and Coordinate Ascent), we use a standard IR metrics nDCG@10 [58] as the main metric to be optimized by our learners.

5.2.2 Results

The first set of experiments examines the application of different feature categories. We include three types of experiments. First, we train each of the LtR algorithms described above varying the feature set used in each experiment (i.e., train with either LETOR, query features, or clinical concepts). Then, we train all learning algorithms using all the 74 features. Lastly, we train all learning algorithms while applying our feature selection method WGAS as described in Section 5.1.4 (we consider the whole feature set as an input, and we set $k = 5$, and $c = 0.01$). In all of these experiments, we compare the learned ranking models to a traditional baseline search system that relies solely on the ranking by BM25 model [97]. We use nDCG@10 [58] and P@10 [30], standard Information Retrieval (IR) metrics, as our main measures to evaluate search results from these different ranking models. All results are shown in Table 5.5; we explain them in detail below.

5.2.2.1 Training with Partial Features

The first three groups of results in Table 5.5 (after the BM25 baseline) present results from training using each group of features separately. All three feature sets have a clear effect on performance, with LETOR features working most effectively and clinical concept features working least effectively. The query features are the least consistent, having a positive effect on some models and a negative effect on others. LETOR features are more consistent, hurting effectiveness only for RankBoost, and also give the greatest and most statistically significant gains in effectiveness; in particular, for Coordinate Ascent, the LETOR features give up to 18% increase over the baseline which is statistically significant with $p < 0.001$ for both nDCG@10 and P@10. This result suggests that in addition to previous studies which indicate the significance of LETOR features in search tasks in other domains [72, 91], LETOR features seem to also be very important in learning models for ranking documents in clinical search tasks.

Method	**Features**	**nDCG@10**	**%Δ**	**P@10**	**%Δ**
Baseline (BM25)	N/A	0.2107	–	0.2844	–
RankBoost (RB)	LETOR	0.1931	-8.3%	0.2700	-5.1%
AdaRank (AR)	LETOR	0.2252	+6.9%	0.2989	+5.1%
LambdaMART (LT)	LETOR	0.2300	+9.1%	0.2700	-5.1%
Random Forests (RF)	LETOR	0.2489**	+18.1%	0.3033	+6.6%
Coordinate Ascent (CA)	LETOR	$\underline{0.2492}^{***}$	+18.3%	$\underline{0.3344}^{***}$	+17.6%
RankBoost (RB)	Query features	0.1925	-8.6%	0.2600	-8.6%
AdaRank (AR)	Query features	0.1893	-10.2%	0.2388	-16.0%
LambdaMART (LT)	Query features	0.2031	-3.6%	0.2580	-9.3%
Random Forests (RF)	Query features	0.2359*	+13.7%	0.2911	+2.3%
Coordinate Ascent (CA)	Query features	$\underline{0.2470}^{**}$	+17.22%	$\underline{0.3067}^{*}$	+7.8%
RankBoost (RB)	Clinical concepts	0.1073	-49.1%	0.1467	-48.4%
AdaRank (AR)	Clinical concepts	0.0390	-81.5%	0.0555	-80.5%
LambdaMART (LT)	Clinical concepts	0.0970	-54.0%	0.1467	-48.4%
Random Forests (RF)	Clinical concepts	0.1057	-50.0%	0.1511	-46.9%
Coordinate Ascent (CA)	Clinical concepts	$\underline{0.1301}$	-38.2%	$\underline{0.1666}$	-41.4%
RankBoost (RB)	All features	0.1970	-6.5%	0.2855	+0.4%
AdaRank (AR)	All features	0.2396*	+13.7%	0.2955	+4.0%
LambdaMART (LT)	All features	0.2222	+5.5%	0.2922	+2.7%
Random Forests (RF)	All features	$\underline{0.2516}^{**}$	+19.4%	$\underline{0.3167}^{*}$	+11.3%
Coordinate Ascent (CA)	All features	0.2416*	+14.7%	0.3011	+5.9%
RankBoost (RB)	Feature selection (WGAS)	0.2224	+5.5%	0.3066	+7.4%
AdaRank (AR)	Feature selection (WGAS)	0.2415*	+14.6%	0.3122*	+9.8%
LambdaMART (LT)	Feature selection (WGAS)	0.2620***	+24.3%	0.3111*	+9.4%
Random Forests (RF)	Feature selection (WGAS)	0.2518**	+19.5%	0.3155*	+10.9%
Coordinate Ascent (CA)	Feature selection (WGAS)	$\underline{\mathbf{0.2701}}^{***}$	+28.1%	$\underline{\mathbf{0.3433}}^{***}$	+20.1%
Median of TREC CDS	N/A	0.2124	–	0.2789	–

Table 5.5: Performance results of applying LtR learners to medical search tasks using various feature categories. Statistical significance (one sided paired t-test) over the baseline (BM25) is denoted by * for $p < 0.05$, ** for $p < 0.01$, and *** for $p < 0.001$. Bold-faced values represent the approach with maximum performance. Underlined values represent the approaches with highest performance for each feature category.

5.2.2.2 Training with Full Features

The next group of results in Table 5.5 shows results from training with all three feature sets. The results are more inconsistent than any of the three blocks above. Coordinate Ascent, which performs best for both measures and all three feature sets alone, has lower effectiveness when all features are included. AdaRank and LambdaMART experience a performance increase for one metric and a decrease for the other compared to their best performance in the three groups above (they still outperform the baseline in both cases). Random Forests, however, do better than their best result in the three individual groups. It is likely that this inconsistency is due to model overfitting for some learners, especially considering that our data set is relatively small (only 90 queries) compared to the total number of features.

5.2.2.3 Training with Feature Selection

The final group of results in Table 5.5 shows results using WGAS algorithm. We note that our approach gives consistent, substantial improvement over using no feature selection at all, resulting in nearly every learning approach having statistically significant improvements over the baseline. LambdaMart and Coordinate Ascent experience the greatest gains for NDCG@10, though as above LambdaMart has inconsistent results for P@10. Moreover, the results for Coordinate Ascent with our feature selection approach are the best of all reported.

Because WGAS works by adding only features that maximize a learned ranking model, a small portion of features that achieve a significant gain will be selected after examining all 74 features–in this case, 19 features for both RankBoost and Random Forests, 13 for AdaRank, 15 for LambdaMART, and 14 for Coordinate Ascent. Moreover, by looking into our training data, we notice that most of those resulting subsets contain ensembles of features that are selected from all three feature categories, which suggests that each of the categories are important in learning effective ranking models for clinical search. In the following section, we provide some analysis of feature importance, and we present top features for each category.

5.2.3 Evaluating Feature Importance

We evaluate the importance (i.e., relevance) of every single feature used in our training. In Table 5.6, we report top features for each of the three feature categories ranked by their weights. We estimate the weight of each feature by computing the average nDCG@10 achieved by learning a model for that single feature which is normalized by the sum of all features' weights.

Unsurprisingly, weighting models such as LM, BM25, and TFIDF, which estimate the relevance of an article to a given query, are shown to be the most effective features in our set suggesting that they are essential in many LtR learning tasks. Furthermore, it seems that the importance of these features is more increased as they are used for larger text content (e.g., full articles) rather than short one (e.g., abstracts and titles). For query features, we observe that some statistics which measure generality (e.g., Scope and AvgIDF) and informative content (e.g., Gamma2 and AvgPMI) [51, 50] seem to be more important in learning LtR models than other statistics within the same category. Lastly, although concept features have the least importance among the three categories, some features (such as counting the total number of matched UMLS concepts in a query and an article, and those concepts belonging to semantic types like "[Sign or Symptom]") are expected to be useful in learning models for medical search tasks.

5.2.4 Evaluating Feature Selection

Another set of experiments we perform is for evaluating our FS method, WGAS, and comparing it to other filter based methods such as XGAS, NGAS [44], and GAS [42]. In these experiments, we examine the application of FS methods using two LtR learning algorithms LambdaMart (pairwise) and Coordinate Ascent (listwise) because they have shown to be the best-performing learners when applying FS as discussed in Section 5.2.2.3. Table 5.7 presents the results of our experiments for Coordinate Ascent and LambdaMART. We use nDCG@10 as our main evaluation metric, and we report the results of the models learned on the subsets of 5%, 10%, 15%, 20%, 25%,

Feature	**Category**	**Weight**
LM model score for the whole article	LETOR	0.0244
BM25 model score for the whole article	LETOR	0.0225
LM model score for `<BODY>` filed	LETOR	0.0209
TFIDF model score for the whole article	LETOR	0.0193
TFIDF model score for `<BODY>` field	LETOR	0.0181
BM25 model score for `<BODY>` field	LETOR	0.0179
IDF estimation for the whole article	LETOR	0.0171
IDF estimation for `<BODY>` field	LETOR	0.0166
TFIDF model score for `<ABSTRACT>` field	LETOR	0.0157
LM model score for `<ABSTRACT>` field	LETOR	0.0153
Gamma2 performance predictor	Query features	0.0201
Query Scope performance predictor	Query features	0.0192
Average inverse document frequency	Query features	0.0188
Average point-wise mutual information	Query features	0.0180
QC1 performance predictor	Query features	0.0175
Sum of all matching concepts	Clinical concepts	0.0095
# concepts in “[Sign or Symptom]”	Clinical concepts	0.0076
# concepts in “[Intellectual Product]”	Clinical concepts	0.0062
# concepts in “[Disease or Syndrome]”	Clinical concepts	0.0061
# concepts in “[Finding]”	Clinical concepts	0.0059

Table 5.6: Top features for each category ranked by weight. The weight of a feature is computed by the average nDCG@10 score achieved by learning a model for the single feature normalized by the sum of all features’ weights.

30%, 35%, 40% of the total number of features available. Also, we set the parameters c and p in GAS and XGAS methods similar to what is shown in [44, 42].

From Table 5.7, we see that applying filter-based feature selection with GAS, NGAS, and XGAS leads to inconsistent results. In particular, we note that in many cases, adding more features to the initially selected subsets do not improve the performance of learned models, but it leads to performance reduction. This could be addressed to the fact that these methods only rely on relevance and similarity statistics for single features and do not maximize a learned model while selecting feature subsets. WGAS, on the other hand, is shown to be a bit more consistent for both learning algorithms and on average it outperforms the three FS methods. This result suggests that although wrapper methods are known to be computationally more expensive, such computational overhead can be tolerated especially when the learning task is performed at small scale (i.e., with few training instances).

Coordinate Ascent (CA)				
Subset (%)	**WGAS**	**NGAS**	**XGAS**	**GAS**
	$c = 0.01$		$p = 0.05$	$c = 0.01$
Full (No FS)	0.2416	0.2416	0.2416	0.2416
5%	0.2517	0.2532	0.2514	0.2560
10%	0.2459	0.2414	0.2501	0.2473
15%	0.2593	0.2329	0.2449	0.2526
20%	0.2593	0.2321	0.2504	0.2485
25%	0.2593	0.2480	0.2498	0.2495
30%	0.2488	0.2418	0.2455	0.2430
35%	0.2558	0.2470	0.2461	0.2401
40%	0.2696	0.2387	0.2463	0.2308
Average	0.2562	0.2418	0.2480	0.2459
LambdaMART (LT)				
Subset (%)	**WGAS**	**NGAS**	**XGAS**	**GAS**
	$c = 0.01$		$p = 0.05$	$c = 0.01$
Full (No FS)	0.2222	0.2222	0.2222	0.2222
5%	0.2105	0.2324	0.2103	0.2176
10%	0.2151	0.2350	0.2200	0.2171
15%	0.2326	0.2136	0.2116	0.2565
20%	0.2317	0.2040	0.2182	0.2262
25%	0.2067	0.1992	0.2494	0.2328
30%	0.2405	0.2132	0.2250	0.2330
35%	0.2431	0.2077	0.2442	0.2222
40%	0.2431	0.2167	0.2067	0.2153
Average	0.2279	0.2136	0.2232	0.2274

Table 5.7: Performance of WGAS, NGAS, XGAS, and GAS algorithms trained with both Coordinate Ascent and LambdaMART reported using nDCG@10.

5.2.5 Discussion And Further Statistical Analysis

Looking at the results from previous sections, one can see that LtR algorithms can be very useful in satisfying user search needs by producing effective search results which match a certain medical case query. Also, compared to a baseline ranking model such as BM25, we observe a major performance improvement suggesting that LtR techniques are essential in the context of clinical decision support. Furthermore, the performance of LtR can be further improved as additional techniques like feature selection are integrated to optimize the learning procedure. However, in terms of effectiveness, we observe that the performance of the resulting ranking model might be tied to which LtR algorithm is used to learn that model. More specifically, in our experiments as described in Section 5.2.2, we see that among the five algorithms, Coordinate Ascent, LambdaMart, and random forests seem to fit our training data better than the other algorithms for most of the cases. In contrast, RankBoost and AdaRank show the lowest performance gains among all the five algorithms.

Thus, we perform analysis of our results to validate that there is a significant performance difference between those five algorithms when used to learn models for clinical search. First, we evaluate whether the difference between those learners (as a group) is statistically significant. Using nDCG@10 as the main evaluation metric, we apply two-way ANOVA to each group of learners used to learn each set of features (we exclude the case of learning with only clinical concepts because none of the learners outperforms the baseline). The last line of Table 5.8 summarizes the results of this analysis. We can clearly see that there is indeed a statistically significant difference between those learners (i.e., $p < 0.01$ when training with query features while $p < 0.05$ for the other feature settings and WGAS feature selection), suggesting that there is a major impact on search results due to the algorithm used to learn a clinical based ranker. In other words, we confirm that there is indeed a statistically significant difference between those algorithms whether they are used with FS or not.

In addition, we evaluate each pair of learners using paired one-sided t-test in order to indicate which learner seems to significantly outperform the other learning algorithms. From the upper part of Table 5.8, we see that both Coordinate Ascent and random forests significantly outperform the other learning algorithms on most of the training cases, which suggests that they might be seen as the best performing algorithms for this task (Coordinate Ascent significantly outperforms random forests only on one instance where WGAS feature selection algorithm is applied). From this analysis, we may conclude that the choice of the learning algorithm, as well as other factors such as the features which are used and the way of selecting those features (i.e., FS method), play a major effect on the overall performance of LtR learning for medical search tasks.

	Statistical Significance			
	FS with WGAS	**All Features**	**Query Features**	**LETOR**
RankBoost (RB)	—	—	—	—
AdaRank (AR)	—	>RB*	—	—
LambdaMART (LT)	>RB**	—	—	—
Random Forests (RF)	>RB**	>RB**, LT*	>RB*, AR**, LT*	>RB**
Coordinate Ascent (CA)	>RB**, AR**, RF*	>RB*	>RB***, AR***, LT**	>RB**, AR*
ANOVA Test	$p < 0.05$	$p < 0.05$	$p < 0.01$	$p < 0.05$

Table 5.8: Statistical Results for using pairwise one-sided t-test to evaluate if the difference between each pair of LTR learners is statistically significant (> denotes a statistical significance over a learner with * for $p < 0.05$, ** for $p < 0.01$, while *** for $p < 0.001$). Also, results for two-way ANOVA test to indicate if there is a significant difference between a group of learners when trained using the same set of features are included in last row.

5.3 Conclusions and Future Work

We presented a framework for applying LtR approaches to clinical search tasks. Our framework includes an optimization technique that can be used to select the most promising feature from a large features pool. Our results show that while using our framework to rank documents for clinical queries, the performance of retrieval systems can be improved to reach up to 28% over a baseline system. Our future work will include applying a similar analysis to the search tasks for other applications such as general health. We aim to generalize these optimization and learning techniques which can result in improved search results for those tasks.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this section, we conclude the thesis by summarizing our main contributions. We have developed and examined several approaches to promote the effectiveness of search results for the clinical domain and to reduce the overhead of manual configurations involved in the retrieval strategies for clinical search. Mainly, we explored three types of approaches that are based on query reformulations, learning concept ratings, and applying learning to rank. They all are shown to be successful in increasing the effectiveness of search results and achieving high performance levels. Specifically, these approaches, which rely on either query reformulation techniques or learning to rank techniques, are shown to achieve substantial retrieval performance compared to strong baseline search systems.

For applying query reformulations on clinical queries, we explored two types of methods to increase retrieval performance. First, we examined the feasibility of using some text analysis techniques such as semantic mapping to gain further understanding of medical terminologies that appear in clinical cases. Using the resulted semantic analysis produced by a semantic mapping utility like MetaMap and by associating each of the mapped concepts to different categories of UMLS types, we were able to distinguish the importance of each concept, and therefore, apply a set of suitable query modification tasks. We proposed a variety of query modification and refinements, ranging from eliminating non-relevant query terms to increasing the weights of terms and concepts that are deemed very important.

Second, we examined the application of ontology-based term expansion to reduce the ambiguity of the terms used in clinical cases. We proposed a novel approach for expanding terms using their equivalent names from MeSH hierarchical resource that is based on disjunctive shared information (DiShIn). Our approach benefits from the directed acyclic graph (DAG) structure of MeSH to overcome some of the limitations in prior work such as over-expanding MeSH terms. This is achieved by using the DiShIn model to compute a distance score between concepts, which can be then used to set a threshold for limiting very broad MeSH concepts from being over-expanded with very narrow concepts.

For concept based rating and learning to perform query reformulations, we proposed a novel method to reduce the manual effort involved in designing and setting query reformulation and modification strategies. Our approach can transform the manual query modifications performed by physicians into automatically learned models using the feature space of UMLS semantic types. We applied the resulted models from this learning approach to a sample of clinical queries and showed how it could effectively predict suitable reformulation action for new unseen query concepts. Our method has the potential of being further explored and generalized for a variety of clinical text understanding tasks.

For applying learning to rank (LtR) techniques on clinical search tasks, our contributions take two directions. First, we explored a comprehensive set of training features from three categories, namely general IR statistics, query performance statistics, and statistics specific to clinical concepts in document-query pairs. Also, we studied the impact of a variety of learning strategies (e.g., boosting, tree-based) as well as different loss functions (pointwise, pairwise, and listwise) when trained using the proposed set of features by including several LtR algorithms such as Random Forests, RankBoost, and Coordinate Ascent. Secondly, to improve the learning process and to limit the dangers of model overfitting, we proposed WGAS, a wrapper method for feature selection that is based on greedy search. This method relies on a combination of feature property statistics and model performance measures to select the set of features which maximize importance and minimize similarities among all features.

We applied our method during the stages of learning LtR models and showed that indeed it could effectively boost the performance of the resulted ranking model, outperforming the state-of-the-art feature selection methods such as GAS, NGAS, and XGAS. In overall, we confirmed the significance of learning to rank methods for promoting the search tasks for clinical and biomedical domain especially when some optimization techniques such as feature selection is included.

6.2 Future Work

In addition to examining the potential of generalizing the approaches proposed in this work to other clinical text mining and understanding tasks, our future work can move in two directions:

1. Exploring the feasibility of performing adaptive clinical search to select the suitable search strategy for each search task,
2. Including user studies and exploring the personalization aspects of clinical search.

6.2.1 Adaptive Clinical Decision Support

Though we developed several search approaches for clinical search tasks and showed that they could lead to significant improvement in the retrieval effectiveness, there are some issues we need to address especially those related to combining the proposed search approaches and using them in a single search system. As discussed in Chapter 3, it seems that the traditional methods for rank fusion, used to aggregate search results for several IR strategies, are less effective when applied to combine our different search strategies. In particular, these methods are shown to result in retrieval effectiveness which tends to be an average point of the performance of the combined search approaches.

We plan to tackle this limitation by exploring the feasibility of applying adaptive clinical decision support which works by evaluating different search strategies on a single query, and then selecting the strategy which is predicted to be the most suitable for that query (i.e., search systems will adaptively choose the best retrieval approach for each clinical search task). This task can be accomplished by using the advances in machine learning (ML) to learn models which can combine two kinds of features, a set which captures some performance properties about the submitted queries [51, 133] (e.g., length, clarity, scope, similarity, etc.) and another set of system features [95] (e.g., indexing strategy, IR model, and expansion model) which distinguishes different search strategies from each other, and then use these models to predict the search strategy which can lead to the highest retrieval performance for a given search task.

Table 6.1 shows an illustration of how to use the proposed learning framework in estimating the performance of the three retrieval approaches discussed in this thesis for a given clinical case. Our future work will focus on implementing the discussed framework using a variety of learning methods (e.g., SVMs, neural networks, random forests, etc.) as well as examining the usefulness of this framework for answering a variety of clinical search tasks such as those for clinical decision support and precision medicine.

	Learning Features			
Clinical Case	**Query Features**	**System Features**	**Strategy**	**Estimated Performance**
Young adult woman with 2 weeks of fever and migrating joint inflammation.	F_1, F_2, F_3,...., F_n	S_1, S_2, S_3,...., S_n	Refinements	0.90
Young adult woman with 2 weeks of fever and migrating joint inflammation.	F_1, F_2, F_3,...., F_n	S_1, S_2, S_3,...., S_n	MeSH	0.88
Young adult woman with 2 weeks of fever and migrating joint inflammation.	F_1, F_2, F_3,...., F_n	S_1, S_2, S_3,...., S_n	LtR	0.55

Table 6.1: Illustration of using the framework to estimate the retrieval performance of Refinements, MeSH, and LtR for a particular clinical case. In this case, "Refinements" is estimated to result in the highest effectiveness compared to "MeSH" and "LtR".

6.2.2 Personalized Clinical Search

Another potential research direction to consider is exploring the usefulness of including some personalization aspects while performing clinical search tasks. Due to the lack of the studies focusing on clinical professionals experience and personalization when performing search tasks, we want to explore different research questions such as validating if leveraging some users' search history data (e.g., the user clicks, query logs) can be used to improve current search results. Also, because clinical professionals represent a wide range of specialties within the medical domain, we want to explore if there is a diversity between those professionals in regards to the type of clinical data that is favored by each group (i.e., the kind of relevant data to the clinicians in the same medical area). Moreover, because in our approaches, we relied on simulated clinicians feedback in selecting relevant biomedical terminologies that appear in clinical queries, we want to conduct a study to capture the actual (i.e., real) feedback provided by those clinicians and compare it to our simulated models.

To accomplish these objectives, we need to turn the prototype search strategies described in this work into production systems and put them on a real working environment. This includes designing a suitable graphical user interface that satisfies user needs and considering other system efficiency aspects to enhance user experience. To succeed in this experiment, we should collect some parameters that indicate the usefulness of our system, such as the time spent during a search session, implicit user feedback (e.g., clicks), user query logs, and any other useful data that is related to each of the user study that is conducted. We expect that the outcomes from these different experiments to help us in a better understanding of the clinicians' behavior while using search systems in making clinical decisions and in enhancing the experience of those practitioners when performing future search tasks.

BIBLIOGRAPHY

[1] Mohammad Alsulmi and Ben Carterette. Examining semantic based query reformulations for clinical based retrieval: Udel at trec cds 2016. In *Proc of TREC*, 2016.

[2] Mohammad Alsulmi and Ben Carterette. Improving clinical case search using semantic based query reformulations. In *Proc of IEEE BIBM*, 2016.

[3] Mohammad Alsulmi and Ben Carterette. Improving medical search tasks using learning to rank. In *Proc of IEEE CIBCB*, 2018.

[4] Mohammad Alsulmi and Benjamin Carterette. Exploring ontology expansions for clinical case retrieval in precision medicine: Udel at trec 2017 pm track. In *Proc of TREC*, 2017.

[5] Mohammad Alsulmi and Benjamin Carterette. Learning to rate clinical concepts using simulated clinician feedback. In *Proc of ACM IUI*, 2017.

[6] Mohammad Alsulmi, Karankumar Sabhnani, and Ben Carterette. University of delaware at trec 2015. In *Proc of TREC*, 2015.

[7] Mohammad R. Alsulmi. Modeling mapreduce workloads on different storage and file system technologies at the extreme scale. Technical report, University of Delaware, 2014.

[8] Giambattista Amati. *Probability models for information retrieval based on divergence from randomness.* PhD thesis, Glasgow University, 2003.

[9] Gianni Amati and Cornelis Joost Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389, 2002.

[10] A. Anand, X. H. Peng, and R. Haywood. Efficient information retrieval via proxy caching mechanisms within a lossy environment. In *Proc of ICITST*, 2009.

[11] Vo Ngoc Anh and Alistair Moffat. Pruned query evaluation using pre-computed impacts. In *Proc of ACM SIGIR*, 2006.

[12] Alan R Aronson and François-Michel Lang. An overview of metamap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236, 2010.

[13] Shivnath Babu. Towards automatic optimization of mapreduce programs. In *Proc of ACM SoCC*, 2010.

[14] Jing Bai, Ke Zhou, Guirong Xue, Hongyuan Zha, Gordon Sun, Belle Tseng, Zhaohui Zheng, and Yi Chang. Multi-task learning for learning to rank in web search. In *Proc of ACM CIKM*, 2009.

[15] Adam Berger and John Lafferty. Information retrieval as statistical translation. In *Proc of ACM SIGIR*, 1999.

[16] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[17] Andrei Z Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. Efficient query evaluation using a two-level retrieval process. In *Proc of ACM CIKM*, 2003.

[18] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. Automatic query expansion using smart: Trec 3. In *Proc of TREC*, 1995.

[19] Chris Buckley and Ellen M. Voorhees. Evaluating evaluation measure stability. In *Proc of ACM SIGIR*, 2000.

[20] Chris Buckley and Ellen M. Voorhees. Retrieval evaluation with incomplete information. In *Proc of ACM SIGIR*, 2004.

[21] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proc of ICML*, 2005.

[22] Mike Cafarella and Doug Cutting. Building nutch: Open source search. *ACM Queue*, 2(2):54–61, 2004.

[23] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Pro of ICML*, 2007.

[24] Ben Carterette, Evangelos Kanoulas, Mark Hall, Ashraf Bah, and Paul Clough. Overview of the trec 2013 session track. In *Proc of TREC*, 2013.

[25] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proc of ACL*, 1996.

[26] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[27] Rudi L Cilibrasi and Paul Vitanyi. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383, 2007.

[28] Charles LA Clarke, Nick Craswell, Ian Soboroff, and Ellen M Voorhees. Overview of the trec 2011 web track. In *Proc of TREC*, 2011.

[29] Charles LA Clarke, Falk Scholer, and Ian Soboroff. The trec 2005 terabyte track. In *Proc of TREC*, 2005.

[30] Paul Clough and Mark Sanderson. Evaluating the performance of information retrieval systems using test collections. *Information Research*, 18(2), 2013.

[31] Alexandru Corlan. *Medline trend: automated yearly statistics of PubMed results for any query*, 2018 (accessed March 28, 2018). http://dan.corlan.net/medline-trend.html.

[32] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proc of ACM SIGIR*, 2006.

[33] Francisco M Couto and Mário J Silva. Disjunctive shared information between ontology concepts: application to gene ontology. *Journal of biomedical semantics*, 2(1):1, 2011.

[34] Francisco M Couto, Mário J Silva, and Pedro M Coutinho. Measuring semantic similarity between gene ontology terms. *Data & knowledge engineering*, 61(1):137–152, 2007.

[35] Koby Crammer and Yoram Singer. Pranking with ranking. In *Proc of Advances in neural information processing systems*, 2002.

[36] W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*, volume 283. Addison-Wesley Reading, 2010.

[37] Van Dang and Bruce Croft. Feature selection for document ranking using best first search and coordinate ascent. In *Proc of SIGIR workshop on feature generation and selection for information retrieval*, 2010.

[38] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[39] Thomas Demeester, Dolf Trieschnigg, Dong Nguyen, and Djoerd Hiemstra. Overview of the trec 2013 federated web search track. In *Proc of TREC*, 2013.

[40] Edward A Fox and Joseph A Shaw. Combination of multiple searches. In *Proc of TREC*, 1994.

[41] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

[42] Xiubo Geng, Tie-Yan Liu, Tao Qin, and Hang Li. Feature selection for ranking. In *Proc of ACM SIGIR*, 2007.

[43] Fredric C Gey and Douglas W Oard. The trec-2001 cross-language information retrieval track: Searching arabic using english, french or arabic queries. In *Proc of TREC*, 2001.

[44] Andrea Gigli, Claudio Lucchese, Franco Maria Nardini, and Raffaele Perego. Fast feature selection for learning to rank. In *Proc of ACM on ICTIR*, 2016.

[45] Veronica Gil-Costa, Mauricio Marin, and Nora Reyes. Parallel query processing on distributed clustering indexes. *Journal of Discrete Algorithms*, 7(1):3–17, 2009.

[46] Aarti Gupta and Tim Oates. Using ontologies and the web to learn lexical semantics. In *Proc of IJCAI*, 2007.

[47] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[48] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[49] David J Hand and Robert J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001.

[50] Claudia Hauff, Leif Azzopardi, and Djoerd Hiemstra. The combination and evaluation of query performance prediction methods. In *Proc of ECIR*, 2009.

[51] Ben He and Iadh Ounis. Query performance prediction. *Information Systems*, 31(7):585–594, 2006.

[52] J He, M Larson, et al. Using coherence-based measures to predict query difficulty. In *Proc of ECIR*, 2008.

[53] Bruce Hedin, Stephen Tomlinson, Jason R Baron, and Douglas W Oard. Overview of the trec 2009 legal track. In *Proc of TREC*, 2009.

[54] Steffen Heinz and Justin Zobel. Efficient single-pass index construction for text databases. *Journal of the Association for Information Science and Technology*, 54(8):713–729, 2003.

[55] Djoerd Hiemstra. A linguistically motivated probabilistic model of information retrieval. *Research and advanced technology for digital libraries*, pages 515–515, 1998.

[56] F. Hu, W. Danny, Q. Mei, and V. Vydiswaran. Learning from medical summaries: The university of michigan at trec 2015 clinical decision support track. In *Proc of TREC*, 2015.

[57] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. Real life, real users, and real needs: A study and analysis of user queries on the web. *Information Processing Management*, 36(2):207–227, January 2000.

[58] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[59] Dawei Jiang, Beng Chin Ooi, Lei Shi, and Sai Wu. The performance of mapreduce: An in-depth study. In *Proc of the VLDB Endowment*, 2010.

[60] Anjali Ganesh Jivani et al. A comparative study of stemming algorithms. *Int. J. Comp. Tech. Appl*, 2(6):1930–1938, 2011.

[61] K Sparck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information processing & management*, 36(6):809–840, 2000.

[62] Karthik Kambatla, Abhinav Pathak, and Himabindu Pucha. Towards optimizing hadoop provisioning in the cloud. In *Proc of Hot Topics in Cloud Computing Conference*, 2009.

[63] Arlind Kopliku, Karen Pinel-Sauvagnat, and Mohand Boughanem. Aggregated search: A new information retrieval paradigm. *ACM Computing Surveys (CSUR)*, 46(3):41, 2014.

[64] Praveen Kumar Lakkimsetti. A framework for automatic optimization of mapreduce programs based on job parameter configurations. Master's thesis, Kansas State University, August 2011.

[65] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[66] Arash Habibi Lashkari, Fereshteh Mahdavi, and Vahid Ghomi. A boolean model in information retrieval for search engines. In *Proc of IEEE ICIME*, 2009.

[67] Victor Lavrenko and W Bruce Croft. Relevance based language models. In *Proc of ACM SIGIR*, 2001.

[68] Hang Li. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862, 2011.

[69] Ping Li, Qiang Wu, and Christopher J Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Proc of Advances in neural information processing systems*, 2008.

[70] Nut Limsopatham, Craig Macdonald, and Iadh Ounis. Inferring conceptual relationships to improve medical records search. In *Proc of OAIR*, 2013.

[71] Nut Limsopatham, Craig Macdonald, and Iadh Ounis. A task-specific query and document representation for medical records search. In *Proc. of ECIR*, 2013.

[72] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[73] Craig Macdonald, Richard McCreadie, Rodrygo LT Santos, and Iadh Ounis. From puppy to maturity: Experiences in developing terrier. In *Proc of OSIR at SIGIR*, 2012.

[74] Craig Macdonald and Iadh Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In *Proc of ACM CIKM*, 2006.

[75] Craig Macdonald, Rodrygo LT Santos, and Iadh Ounis. On the usefulness of query features for learning to rank. In *Proc of ACM CIKM*, 2012.

[76] Craig Macdonald, Rodrygo LT Santos, and Iadh Ounis. The whens and hows of learning to rank for web search. *Information Retrieval*, 16(5):584–628, 2013.

[77] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[78] Richard McCreadie, Craig Macdonald, and Iadh Ounis. Comparing distributed indexing: To mapreduce or not? In *Proc of LSDS workshop of SIGIR*, 2009.

[79] Richard McCreadie, Craig Macdonald, and Iadh Ounis. Mapreduce indexing strategies: Studying scalability and efficiency. *Information Processing & Management*, 48(5):873–888, 2012.

[80] Donald Metzler and W Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.

[81] David RH Miller, Tim Leek, and Richard M Schwartz. A hidden markov model information retrieval system. In *Proc of ACM SIGIR*, 1999.

[82] André Mourão, Flávio Martins, and João Magalhães. Multimodal medical information retrieval with unsupervised rank fusion. *Computerized Medical Imaging and Graphics*, 39:35–45, 2015.

[83] National Library of Medicine. *Fact Sheet: Medical Subject Headings*, 2017 (accessed March 28, 2018). https://www.nlm.nih.gov/pubs/factsheets/mesh.html.

[84] National Institute of Standards and Technology. *Common Evaluation Measures*, 2007 (accessed March 28, 2018). https://trec.nist.gov/pubs/trec16/appendices/measures.pdf.

[85] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proc of ACM SIGIR 2006 Workshop on Open Source Information Retrieval*, 2006.

[86] João R. M. Palotti, Veronika Stefanov, and Allan Hanbury. User intent behind medical queries: An evaluation of entity mapping approaches with metamap and freebase. In *Proc of ACM IIiX*, 2014.

[87] Joao Palotti, Lorraine Goeuriot, Guido Zuccon, and Allan Hanbury. Ranking health web pages with relevance and understandability. In *Proc of ACM SIGIR*, 2016.

[88] Joao Palotti and Allan Hanbury. Tuw@ trec clinical decision support track 2015. In *Proc of TREC*, 2015.

[89] Ari Pirkola and Kalervo Järvelin. Employing the resolution power of search keys. *J. Am. Soc. Inf. Sci. Technol.*, 52(7):575–583, May 2001.

[90] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proc of ACM SIGIR*, 1998.

[91] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374, 2010.

[92] C. J. Van Rijsbergen. *Information Retrieval.* Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[93] Kirk Roberts, Dina Demner-Fushman, Ellen Voorhees, and William Hersh. Overview of the trec 2016 clinical decision support track. In *Proc of TREC*, 2016.

[94] Kirk Roberts, Dina Demner-Fushman, Ellen M Voorhees, William R Hersh, Steven Bedrick, Alexander J Lazar, and Shubham Pant. Overview of the trec 2017 precision medicine track. In *Proc of TREC*, 2017.

[95] Kirk Roberts, Matthew Simpson, Dina Demner-Fushman, Ellen Voorhees, and William Hersh. State-of-the-art in biomedical literature retrieval for clinical cases: a survey of the trec 2014 cds track. *Information Retrieval Journal*, 19(1):113–148, 2015.

[96] Kirk Roberts, Matthew Simpson, Ellen Voorhees, and William Hersh. Overview of the trec 2015 clinical decision support track. In *Proc of TREC*, 2015.

[97] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[98] J. J. Rocchio. A study of smoothing methods for language models applied to ad hoc information retrieval. *The SMART retrieval system experiments in automatic document processing*, 1971.

[99] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the Association for Information Science and Technology*, 41(4):288–297, 1990.

[100] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[101] Mark Sanderson et al. Test collection based evaluation of information retrieval systems. *Foundations and Trends® in Information Retrieval*, 4(4):247–375, 2010.

[102] Eric Sayers. *E-utilities Quick Star*, 2017 (accessed March 28, 2018). https://www.ncbi.nlm.nih.gov/books/NBK25500.

[103] Falk Scholer, Hugh E. Williams, and Andrew Turpin. Query association surrogates for web search: Research articles. *J. Am. Soc. Inf. Sci. Technol.*, 55(7):637–650, May 2004.

[104] Matthew S Simpson, E Voorhees, and William Hersh. Overview of the trec 2014 clinical decision support track. In *Proc of TREC*, 2014.

[105] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[106] Luca Soldaini and Nazli Goharian. Learning to rank for consumer health search: a semantic approach. In *Proc of ECIR*, 2017.

[107] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[108] Trevor Strohman and W Bruce Croft. Efficient document retrieval in main memory. In *Proc of ACM SIGIR*, 2007.

[109] Trevor Strohman, Donald Metzler, Howard Turtle, and W Bruce Croft. Indri: A language model-based search engine for complex queries. In *Proc of International Conference on Intelligent Analysis*, 2005.

[110] Trevor Strohman, Howard Turtle, and W Bruce Croft. Optimization strategies for complex queries. In *Proc of ACM SIGIR*, 2005.

[111] Mario Taschwer. Text-based medical case retrieval using mesh ontology. In *Proc of CLEF Workshop*, 2013.

[112] TREC Precision Medicine track website. *2017 Precision Medicine Track*, 2017 (accessed March 28, 2018). http://www.trec-cds.org/2017.html.

[113] TREC Precision Medicine track website. *Relevance Judgment Guidelines*, 2017 (accessed March 28, 2018). http://www.trec-cds.org/relevance_guidelines.pdf.

[114] Howard Turtle and James Flood. Query evaluation: strategies and optimizations. *Information Processing & Management*, 31(6):831–850, 1995.

[115] Manisha Verma, Emine Yilmaz, Rishabh Mehrotra, Evangelos Kanoulas, Ben Carterette, Nick Craswell, and Peter Bailey. Overview of the trec tasks track 2016. In *Proc of TREC*, 2016.

[116] Ellen M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. In *Proc of ACM SIGIR*, 1998.

[117] Ellen M Voorhees. The trec medical records track. In *Proc of ACM BCB*, 2013.

[118] Ellen M Voorhees and William R Hersh. Overview of the trec 2012 medical records track. In *Proc of TREC*, 2012.

[119] Ellen M Voorhees and Richard M Tong. Draft: Overview of the trec 2011 medical records track. In *Proc of TREC*, 2011.

[120] Raymond Wan, Jannifer Hiu-Kwan Man, and Ting-Fung Chan. Modification through external sources to support clinical decisions. In *Proc of TREC*, 2014.

[121] Kewen Wang, Xuelian Lin, and Wenzhong Tang. Predator- an experience guided configuration optimizer for hadoop mapreduce. In *IEEE CloudCom*, 2012.

[122] Yue Wang, Xitong Liu, and Hui Fang. A study of concept-based weighting regularization for medical records search. In *Proc of ACL*, 2014.

[123] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, 2012.

[124] Ian H Witten, Alistair Moffat, and Timothy C Bell. Managing gigabytes: Compressing and indexing documents and images. *IEEE Transactions on Information Theory*, 41(6):2101, 1995.

[125] Dili Wu. A profiling and performance analysis based self-tuning system for optimization of hadoop mapreduce cluster configuration. Master's thesis, Vanderbilt University, May 2013.

[126] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

[127] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proc of ACM SIGIR*, 2007.

[128] Emine Yilmaz and Javed A Aslam. Estimating average precision with incomplete and imperfect judgments. In *Proc of ACM CIKM*, 2006.

[129] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[130] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proc of ACM SIGIR*, 2001.

[131] Min Zhang, Ruihua Song, Chuan Lin, Shaoping Ma, Zhe Jiang, Yijiang Jin, Yiqun Liu, Le Zhao, and S Ma. Expansion-based technologies in finding relevant and new information: Thu trec 2002: Novelty track experiments. In *Proc of TREC*, 2002.

[132] Weizhe Zhang, Hui He, and Jianwei Ye. A two-level cache for distributed information retrieval in search engines. *The Scientific World Journal*, 2013, 2013.

[133] Y Zhao, F Scholer, and Y Tsegay. Effective pre-retrieval query performance prediction using similarity and variability evidence. In *Proc of ECIR*, 2008.

[134] Yun Zhou and W. Bruce Croft. Query performance prediction in web search environments. In *Proc of ACM SIGIR*, 2007.

[135] Dongqing Zhu and Ben Carterette. Improving health records search using multiple query expansion collections. In *Proc of IEEE BIBM*, 2012.

[136] Dongqing Zhu and Ben Carterette. Joint search in text and concept spaces for emr-based cohort identification. In *Proc of IEEE BIBM*, 2013.

[137] Justin Zobel. How reliable are the results of large-scale information retrieval experiments? In *Proc of ACM SIGIR*, 1998.

Appendix

EFFICIENCY CONSIDERATIONS FOR CLINICAL SEARCH SYSTEMS

Two aspects of performance can be measured for information retrieval systems: the effectiveness of search results (in terms of the quality of search results), and the efficiency of these systems (in terms of the time spent during generating search results). Due to the nature of our research objective, we tend to put more focus on the aspects for increasing the effectiveness of search systems and focus less on the other efficiency aspects such as optimizing disk space usage and reducing processing time. In particular, in Chapter 3, Chapter 4, and Chapter 5, we focused on exploring several IR methodologies that aim to increase the effectiveness of search results that are produced by clinical search systems. Nevertheless, in this part, we consider providing some discussions on how to efficiently design and implement a variety of search and retrieval strategies.

The core of search systems is based on two operations, document indexing, which transforms the representation of documents into posting lists that are more suitable for search and retrieval, and document retrieval, done by computing a similarity score between query and document using the statistics of posting lists. In the following two sections, we will examine some efficiency considerations related to performing these two operations.

A.1 Document Indexing

Search systems rely on an indexing layer to allow efficient retrieval of documents from a given corpus [78]. Typically, when the indexing procedure is performed, search systems create certain types of data structure (e.g., lists, hash-tables, etc.) which can be suitable for future retrieval needs, and together these structures maintain what is known as an index. The main index usually contains an inverted index which consists of posting lists for all terms. Each term will be assigned a single posting list, a list of all documents (identified by document-ID) containing that term [124]. The posting list structure is very beneficial for retrieval purposes as it stores sufficient statistical information to be used for scoring each document, such as the frequency of the term occurrences, positional information of the term within each document, and field information (i.e., the occurrence of the term in different fields of the document, such as title, allowing for boosting some of these fields during retrieval) [78].

The procedure of creating an index for a collection of documents can be described as follows. First, documents are read from the disk and then they are tokenized. Usually, some further processing involves the tokenization step such as applying stemming algorithms and removing stopwods [60, 78]. Then, a single-pass indexing [54] is applied, in that the posting lists for terms are built in the main memory while scanning the corpus. Because it is unlikely to fit the posting lists for all the documents in the main memory (especially when only a single computing node is considered), the partial index will be flushed and written back to the disk. When a full scanning of all the documents in the corpus is performed, the final index will be generated by combing the previously flushed indices, resulting in a single index for the entire corpus [78].

A.1.1 The Importance of Efficient Document Indexing

Similar to search tasks for other domains (e.g., general web search), performing clinical and biomedical search tasks will require relying on efficient indexing which helps in producing search results and then return them to users on time. Using effective ways to build indexes has its importance in IR applications and especially in biomedical search. This is addressed due to the fact that clinical search systems usually make the use of several biomedical resources as well as several document and query representations (e.g., maintaining different indexes for text and concept representations as in Zhu et al. [136] and Limsopatham et al. [71]), which result in performing extra processing and needing to build more document indices. In addition, the massive growth of the number of publications in biomedical literature, reaching about 1.3 million new articles published yearly [31], will require search systems to be continuously updated (i.e., rebuilding and updating the index) in order to keep up with the recent advances in the biomedical research domain.

There are several ways to optimize the indexing procedure which can result in reducing the total time for generating indexes. Many of these approaches can be applied in biomedical search settings to minimize index creation time. In the next few sections, we present a number of these approaches starting by discussing the impact of parallel document indexing, and later, in Section A.1.3, we discuss some other optimization methods that are related to examining different settings of system configurations.

A.1.2 Distributing Indexing Procedure

One potential way to increase the efficiency of document indexing (i.e., in terms of processing time) is by distributing the indexing procedure, described previously, on several processing nodes that are running concurrently. In ideal situations, assuming that the documents to be indexed are partitioned evenly among the processing nodes where those partitions are local to the machines doing the work (no data sharing among those machines), we would expect this process to always result in an optimal scaling (i.e., the speedup will scale linearly with the number of allocated nodes). However, in practical situations, having a fully distributed processing with local data might not be easily achieved. This is true especially when the indexing process is performed at large scale (too many nodes are considered) as we might be overloading the network traffic while distributing the corpus or we might be risking data loss from a single point of failure [78].

Fortunately, MapReduce, a parallel programming model that facilitates the processing of large datasets, provides sufficient solution for distributing our indexing procedure. MapReduce is currently implemented by various platforms such as Hadoop

[123] and Spark [129], and it seems to be appealing to many scientific problems because of its simplicity, automatic load balancing, failure recovery, and ease of scaling nature [38]. The procedure that MapReduce model follows is performed by splitting computational/data processing tasks into two stages: a mapping stage, where the input data is read and transformed to an intermediate representation, followed by a reduction stage, where the intermediate data is gathered and merged into the final desired results. Both stages can be efficiently distributed across multiple machines.

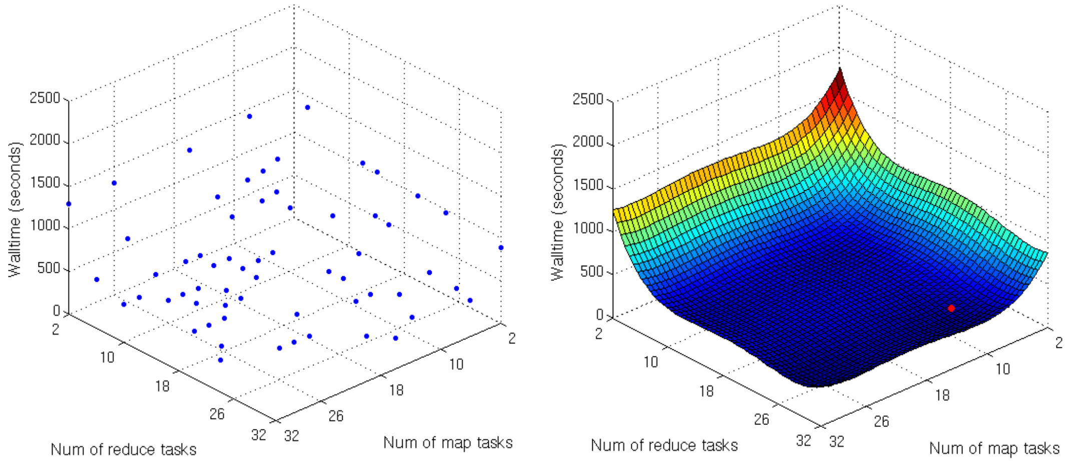
Relying on the abstraction of MapReduce, several approaches have been proposed to perform distributed document indexing, varying the way of fitting the indexing procedure within the map and reduce stages [38, 22, 78, 79]. For instance, Dean et al. [38] described a way for indexing with MapReduce which maps each term and document-ID from the input documents and then applies a reduction step to generate the desired posting lists in the form of `<term, list(document ID)>`. This implementation of distributed indexing is shown to scale very well with an increased number of computing resources; however, one drawback is the performance overhead caused by the large volume of intermediate data generated from the map stage output [78]. Taking a step-forward towards reducing intermediate output and lowering disk and network IO traffic, Mccreadie et al. [78, 79] propose a more efficient way to implement document indexing with MapReduce. The new implementation works by mapping input documents into partial posting lists (i.e., partial indices) and then applying a reduction step which merges partial posting lists for each term and combines them to generate the full index. In fact, this strategy is shown to outperform the approach by Dean et al. and scales close to the optimal linear scaling (this strategy is available on Terrier IR [85], the underlying system we use in our experiments described in Section 3.2.3.1).

A.1.3 The Effect of System Configurations

Besides parallelizing document indexing, system configurations can have a significant impact on the efficiency of the index creation process. Assuming that the indexing is performed using a MapReduce based framework such as Hadoop, several studies have suggested that the selection of that framework's parameters (e.g., allocated memory size, assigned parallel processes, IO block size, etc.) plays a major effect on the overall performance of the running system [13, 59, 62, 64, 125]. As a result, many methods have been proposed to optimize the selection of those configurations and to choose the most suitable parameters for running MapReduce applications [64, 121, 125, 7].

For instance, in the approach by Lakkimsetti [64], the author considered a subset of configuration parameters that are related to the number of parallel processes and the size of I/O sorting buffers. Then, he developed an optimizer that runs an exhaustive search on these parameters to find the best set of values which results in a reduction of processing time. By applying this approach on a diverse set of applications, a set of system performance statistics are collected, and when a new job is submitted, the optimizer categorizes the new job and then predicts that set of parameter values that are best fit for this category of applications.

Another type of approaches performs much efficient exploration of the parameter space, achieved by running optimized search algorithms such as Grid Hill Climbing and Simulated Annealing [121, 125]. In such approaches, the optimized search is applied to a small instance of the entire task (e.g., indexing a subset of documents in the corpus), and then the whole task will be executed with the optimized parameter values. Moreover, aiming to control and reduce the cost of search in the methods described above, in [7], we examined an approach for learning models generated by randomly sub-sampling the whole parameter space. For instance, in Figure A.1, we show an example for learning polynomial surface model for one MapReduce task, PageRank, by sampling two parameters. By using this approach, we can learn accurate models relying on very few points in the space and then use the resulted models to predict configuration parameters that are very close to the optimal values (i.e., we can achieve 95% prediction accuracy of optimal settings by sampling less than 25% of the entire space). The outcomes from these optimizations indicate a significant speedup resulting



(a) Sampled points

(b) Learned model

Figure A.1: Example for learning models for predicting configuration parameters using a sub-sampled space.

from running MapReduce jobs with predicted configuration parameters.

The effect of system configurations is not only limited to the settings of framework and software applications but in fact, it can be extended to include some other system aspects like the type of technology used in the hardware and storage systems. In a previous study described in [7], we looked into the impact of varying file storage systems, while performing different indexing tasks, by considering two types of technologies, solid-state flash drive (SSD) that is locally connected to each computing node, and a hard drive storage (Lustre) that is distributed across a cluster of computing nodes. The results of this experiment are summarized in Figure A.2 using processing time (in seconds) as our main evaluation measure. From Figure A.2, one can see that because indexing tasks usually are IO bound tasks, it seems that the type of technology used in storage system will have a significant efficiency impact. In our case, it is shown that the local SSD outperforms the shared Lustre storage, achieving a speedup up to 2 when experimenting with nutchindexing application.

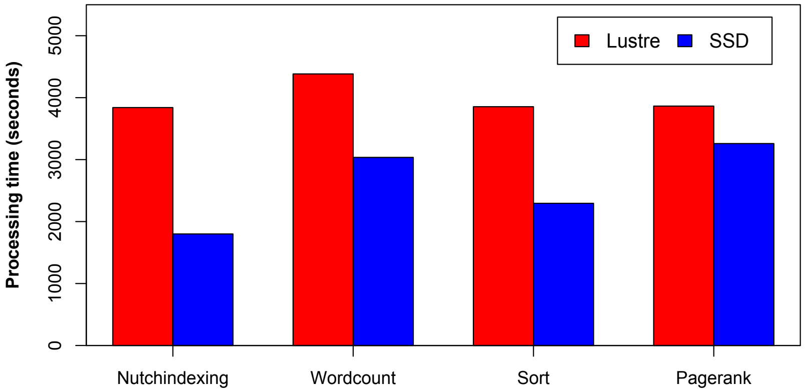


Figure A.2: Comparing two file system technologies with various indexing and IR tasks

A.2 Document Retrieval

The efficiency of document retrieval can be tied to the data structures used in the underlying indexing layer. Assuming that very efficient structures are used to store term and document statistics in the index, additional techniques can be considered to increase the efficiency of query processing and reduce retrieval time. Traditionally, search systems perform retrieval by applying an exhaustive evaluation of query terms. In particular, supposing that term-at-a-time processing strategy is employed, search systems keep track of a set of accumulator variables for all documents containing query terms. Then, each term's inverted lists is thoroughly scanned and processed before moving to the next term, and the partial contributions of term-document similarity scores are added to the accumulators of the corresponding documents [114, 17, 11]. The exhaustive query evaluation will be finalized as soon as the processing of all the terms' inverted lists is completed, and then, a list of k documents with the highest accumulators (e.g., 1000 documents) is returned as a response to the query asked by the user.

A.2.1 The Importance of Efficient Document Retrieval

Enhancing the efficiency of clinical search systems by speeding up query processing and reducing system response time is an important aspect for increasing systems' availability and satisfying users. One major limitation in the exhaustive query evaluation described above is the need for large memory space to keep all the accumulators in addition to more processing time that is required to compute similarity scores. This becomes a major drawback especially when the evaluated queries contain terms that are common across many documents, leading to larger accumulators space as well as more processing time to update all the accumulators [11]. One optimization could be performed by applying some techniques for pruning and inverted list skipping which can significantly reduce processing time without harming the quality of search results [17, 110, 11, 108]. As an example, Anh et al. [11] propose a pruning technique which reduces the size of the document score accumulators by making the use of impact-sorted inverted lists. In such method, document pointers in an inverted list for a given term will be sorted by the impact of that term within them, allowing to skip a large number of documents containing terms with low impact and to save some processing time.

Some other optimization techniques are more related to specific features used in search systems (e.g., query reformulations, term expansion, etc.) or related to particular search domain (e.g., general web, biomedical search). In the next few sections, we discuss some of these methods, focusing more on showing some examples of how these approaches can be applied for clinical and biomedical search.

A.2.2 The Processing of External Biomedial Resources

Generally, search systems make the use of some external resources (e.g., user log files, word dictionaries) to leverage the effectiveness of search results. Notably, in biomedical search, those resources can be used to improve the clarity of queries submitted by users and reduce the ambiguity of biomedical terminologies. Medical Subject Headings (MeSH) [83] is one of the commonly used resources to expand original

Method	**Throughput (desc/sec)**	**Speedup**	**Disk Size (MB)**
MeSH Resource (unfiltered)	1144	1.00	313
MeSH Resource (filtered)	13728	12.00	13

Table A.1: The performance impact of filtering the original MeSH resource file. Throughput represents the number of concept descriptors that are loaded into the memory per second.

query terms with their other equivalent medical vocabularies, which is shown to be very useful in increasing the quality of search results as discussed in Section 3.2.2.3, Section 3.2.3.2, and Section 3.3.4.2.

MeSH resource is provided in a large semi-structured file (in XML format) containing descriptors for about 28,000 medical concepts. Each concept's descriptor lists some detailed attributes about that concept such as its unique ID, heading text, tree numbers, additional terms, annotations, scope note, and some other related information. Because this file is quite large (about 313 MB), it may not be effective to load all the descriptors within the file and manipulate them in the main memory as much more memory space will be dedicated for this task. One potential solution is to filter out some of the unwanted attributes and create an intermediate file containing a filtered set of concept descriptors. For instance, because our MeSH expansion approach only requires four attributes ID, heading, entry terms, and tree numbers, by applying a simple filtering approach, we were able to reduce the original file's size into much smaller one. As presented in Table A.1, it seems that applying such a filtering mechanism affects performance significantly, and in fact, we can parse and load this filtered set into the main memory much faster than the original unfiltered set, which is shown to result in a speedup of 12.

Another optimization, which makes the use of MeSH resource really efficient after the loading of the filtered set into the memory, is achieved by creating data indices and altering the way to access MeSH descriptors. Initially, MeSH resource is organized in a sequential manner (i.e., a list) in that you will need to traverse

Method	**Throughput (desc/sec)**	**Speedup**	**Memory Size (MB)**	**Disk Size (MB)**
Sequential Access + Disk	<1	1.00	<2	8
Sequential Access + Memory	7.22	1.00	40	—
Indexed Access + Disk	1.47	36.75	<2	12
Indexed Access + Memory	204.5	28.40	95	—

Table A.2: Comparing difference strategies for accessing MeSH descriptors. Speedup is computed with respect to to whether disk or memory access is performed.

through the entire descriptors list to locate a certain concept. Our expansion approach, described in Section 3.2.2.3, requires to access MeSH resource frequently in order to expand query concepts and to control the expansion process by estimating semantic similarity between those concepts (i.e., applying steps 1 through 4 in Section 3.2.2.3), so having very efficient access to MeSH will be crucial in answering user queries in a timely manner.

In Figure A.3, we present a proposed strategy for reducing the time needed to access MeSH descriptors by relying on three hash indices. In this strategy, we use one index to access the descriptors using the original concept heading (identified from user query) whereas the two other indices are used to guarantee speedy access to concept descriptors and concept's children trees while performing similarity estimation between concepts. Table A.2 compares the performance of our proposed indexing access to a baseline, sequential descriptor access, considering two scenarios either accessing the entire indices as well as the descriptors from memory or accessing them from the hard disk. In our evaluation, we applied the expansion process on the concepts used in the topic sets for the TREC CDS track in 2014, 2015, and 2016.

From Table A.2, one can see that for both memory and disk access strategy cases, using the indexed structure for retrieving concept descriptors tend to be extremely efficient, and in fact, it is shown to be at least 28 times faster than accessing descriptors sequentially. Of course, we would expect a trade-off between the speedup

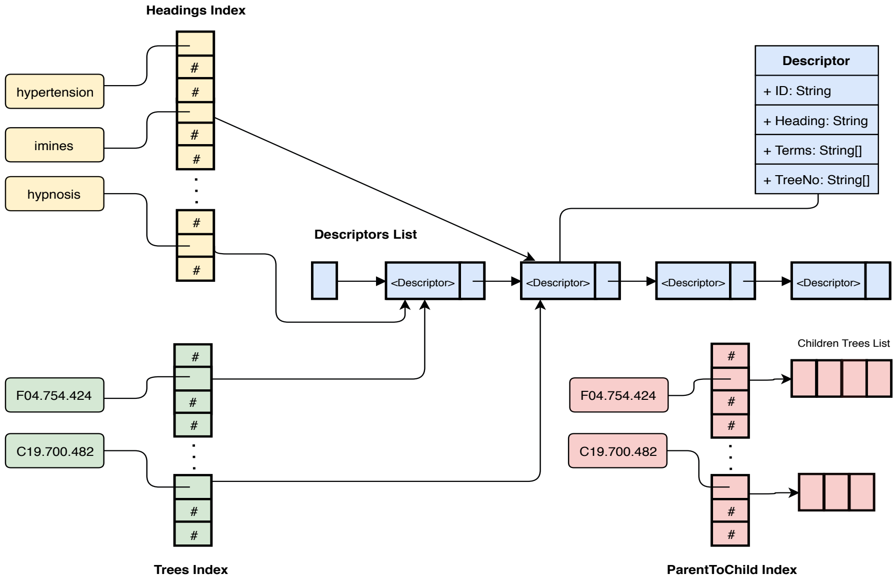


Figure A.3: A structure for speedy access and processing of MeSH concepts using three hash indices.

and the memory usage as more memory space will be allocated for our proposed strategy to store the structure of the additional indices; however, considering the current advances in computer memory sizes, the memory usage of this strategy seems to be within the acceptable range. In overall, in our experiments, we showed that applying different strategies to process external resources indeed can have a noticeable performance impact when performing retrieval tasks.

A.2.3 The Impact of Data Caching

Caching is a simple yet an effective technique for storing frequently used data in memory. So, when the same data is requested again, it can be directly retrieved from the memory without the need to generate that data or to request it from some other resources. Introducing a caching layer in search systems can be extremely important to boost the performance of these systems, as we expect it to enhance system response time by reducing the overhead of a variety of bottleneck resources (e.g., disk storage, network traffic, etc.).

Data caching techniques can be effectively implemented within different levels of search systems. They can be implemented at the user query level by storing search results for the most frequently asked queries, resulting in a major reduction of query processing time as suggested by Zhang et al. [132]. Also, they can be integrated with the system index (by caching a subset of the term's inverted lists) which is expected to save some CPU time when computing query-document similarity scores. Moreover, caching can be applied at the network level to enhance network usage and reduce traffic by storing some frequent network requests that are performed by search systems (i.e., when demanding some services from online resources) [10].

In our work, we rely on caching techniques to reduce system response time and reduce network traffic. Specifically, because our implementation of MeSH uses Entrez online E-utilities [102] to access PubMed in order to estimate the weights of terms after applying expansion (step 5 in Section 3.2.2.3), we build a cache to store the most recent term weight estimation requests (the cache will contain a set of terms and their weights from PubMed). If a new query, which is submitted by a user, contains a set of concepts that are expandable with MeSH terms and have been already included in some recent queries, then the weight estimation of these concepts' expansion terms will be performed using the previously cached data. Note that we cannot eliminate network traffic by storing PubMed locally because such collection is not publicly available. By conducting an experiment (presented in Table A.3) with the set of all expandable concepts in the TREC CDS 2014, 2015, and 2016 topics, our approach is shown to enhance the performance of our search system significantly, resulting in execution time that is three times faster than if no caching is used. From this result, we can see that caching technique can play a significant role in reducing response time and enhancing system performance.

A.2.4 The Impact of Parallel Execution

Besides processing and efficiently caching data from external biomedical resources, parallel task execution can be an effective approach to increase the efficiency

Method	**Throughput (term/sec)**	**Speedup**
Term Weight Estimation (uncached)	6.4	1.00
Term Weight Estimation (cached)	19.2	3.00

Table A.3: The performance impact of caching term weight estimation results over applying no caching. Throughput represents the number of estimated weights for terms per second.

of retrieving search results. Similar to applying parallelism in indexing, addressed in Section A.1.2, performing parallel execution can be as well beneficial in enhancing the performance of document retrieval. In fact, there are several ways to apply parallelism within search systems. One way is by distributing query evaluation procedure among several machines. In this way, the processing of each user query (especially complex queries and those require some additional resources) will be performed using several computing nodes, and the final results will be returned after gathering terms and phrases statistics of that query from all nodes [109]. Another way is accomplished by performing parallelism at user query level, meaning that several queries will be processed concurrently across a large number of machines but without distributing the evaluation of each single query [108].

Several studies have suggested that applying parallel query processing is essential for enhancing the performance of search systems [109, 108, 45]. For instance, Strohman et al. [109] introduced a search system in which concurrent processing is applied to reduce search response time and to improve system throughput. The experiments performed by the authors using a cluster of six machines show that parallel processing is very significant in increasing the efficiency and reducing the time for processing user queries (in this study, a given user query can be processed efficiently within almost a second).

Just like the previous studies, in our work, we examined the effectiveness of concurrently processing a sample of clinical queries. As shown in Table A.4, by relying on the multithreading package in Java and by using a machine with single node and four

Method	**Throughput (query/min)**	**Speedup**
Sequential Execution	22.11	1.00
Parallel Execution	72.58	3.28

Table A.4: The performance impact of applying parallel query execution compared to sequential execution for all the topics in TREC CDS 2014, TREC CDS 2015, and TREC CDS 2016. Throughput represents the number of processed queries per minute.

cores, our experimental results confirm that applying such technique can indeed lead to a speedy query processing with a speedup of over three compared to sequential single core execution (note that the shared memory bandwidth is limiting our parallelism from reaching perfect linear scaling). From these results, we can clearly distinguish the importance of parallel processing in boosting the performance of search systems.

A.2.5 Combining Document Retrieval Optimizations

To show the impact of retrieval optimizations discussed above when they integrated together, we combine all of the three proposed methods (i.e., processing and storing MeSH concepts, caching PubMed data, and applying parallel execution) in one setting and we evaluate their efficiency when applied in clinical search. In our evaluation, we used the topic sets for TREC CDS track for the years 2014, 2015, and 2016 (30 topics for each year).

For each topic set, we perform two experiments. First, we run our retrieval system with all the different components but without applying any of the suggested optimizations. In particular, we use the sequential memory access with the unfiltered set of MeSH descriptors (described in Section A.2.2) to process the set of MeSH terms appearing in the clinical queries, and we apply sequential execution to process these queries (also, we do not perform any caching of term weight information for this case). Second, we run our retrieval system on each set of the provided topics, but in this case, we consider applying the structured MeSH indexing, caching term weight data, and performing parallel execution of all queries. In Table A.5, we summarize the results of

Topic set	**Method**	**Total time (sec)**	**Throughput (query/min)**	**Speedup**
CDS 2014	unoptimized retrieval	442 seconds	4.1	1.0
	optimized retrieval	109 seconds	16.5	4.0
CDS 2015	unoptimized retrieval	457 seconds	3.9	1.0
	optimized retrieval	94 seconds	19.1	4.8
CDS 2016	unoptimized retrieval	521 seconds	3.5	1.0
	optimized retrieval	153 seconds	11.8	3.4

Table A.5: Comparing the efficiency of applying a set of optimization techniques when performing document retrieval over applying no optimizations. Total time represents the processing time of all the topics in each topic set whereas throughput represents the number of processed queries per minute.

our experiments, and we show that combining the set of the proposed enhancements indeed results in a noticeable performance impact leading to a speedup ranging between 3.4 and 4.8. This result confirms the importance of applying different optimization strategies to enhance the efficiency of retrieval and to reduce response time when performing clinical search.

A.2.6 Understanding the Trade-off Between Efficiency and Effectiveness

The discussed optimization techniques are shown to lead to a significant speedup and improve the efficiency of our search system. However, one important aspect we need to consider when developing new search strategies for enhancing the effectiveness of search systems is that, in many cases, these strategies would come with some additional computational costs, leading to slower query processing compared to an essential baseline search strategy. In particular, for clinical search tasks, many of the proposed approaches make the use of some external resources (e.g., MeSH, UMLS, Genetics Home Reference, etc.) in order to reduce the complexity of those tasks and to leverage the performance of search system. Thus, adding these components to a search system would imply performing extra processing to fetch some data from disk or memory and in some other cases relying on a network medium to access and retrieve remotely stored data (e.g., term weights in our MeSH).

Topic set	Method	Throughput (query/min)	P@5	P@10	nDCG@10
CDS 2014	Baseline	70.1	0.3800	0.3500	0.2899
	MeSH expansion	16.5	0.4200	0.3900	0.3511
CDS 2015	Baseline	86.5	0.4467	0.4100	0.2706
	MeSH expansion	19.1	0.4400	0.4033	0.2682
CDS 2016	Baseline	61.3	0.2800	0.2767	0.1964
	MeSH expansion	11.8	0.3467	0.3033	0.2131

Table A.6: The efficiency of clinical search systems is compared to the effectiveness of search results to show a case of trade-off between the two factors. Efficiency is measured using throughput (query processed per minute) whereas effectiveness is measured using P@5, P@10, and nDCG@10.

In fact, we would expect a trade-off between the efficiency of search systems and the effectiveness of the search results which they produce in that to achieve high-quality search results, we will need to sacrifice some efficiency aspects of our system. For instance, looking at the results presented in Table A.6, we see that although we apply some optimizations to enhance the efficiency of our MeSH strategy and this strategy is shown to be highly effective with improvement in search results up to 23%, it seems the resulted efficiency (in terms of query processing throughput) is four to five times slower than our baseline search strategy (i.e., with no expansion). Generally, in practical cases such as when search systems are put in production stages, some further analysis will be needed to decide whether it is worthwhile to tolerate the additional computational costs of the new proposed strategies knowing how much retrieval effectiveness gain they could make.

A.3 Summary and Concluding Remarks

We discussed a wide range of methods related to improving the efficiency of search systems and reducing response time. Mainly, we considered two categories of performance optimization techniques; one group of techniques can be performed during the indexing stage to minimize indexing time and use system resources efficiently, and another which is performed during retrieval to reduce query processing and respond to users on time. Those techniques are shown to apply to a wide range of search tasks in IR, but in our discussion, we focused on showing some examples from the clinical and biomedical domain.