Using a Combination of Methodologies for Improving Medical Information Retrieval Performance

Hoda Forghani Raissi

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF ARTS

GRADUATE PROGRAM IN
INFORMATION SYSTEMS AND TECHNOLOGY
YORK UNIVERSITY
TORONTO, ONTARIO

September 2013

ABSTRACT

This thesis presents three approaches to improve the current state of Medical Information Retrieval. At the time of this writing, the health industry is experiencing a massive change in terms of introducing technology into all aspects of health delivery. The work in this thesis involves adapting existing established concepts in the field of Information Retrieval to the field of Medical Information Retrieval. In particular, we apply subtype filtering, ICD-9 codes, query expansion, and re-ranking methods in order to improve retrieval on medical texts. The first method applies association rule mining and cosine similarity measures. The second method applies subtype filtering and the Apriori algorithm. And the third method uses ICD-9 codes in order to improve retrieval accuracy. Overall, we show that the current state of medical information retrieval has substantial room for improvement. Our first two methods do not show signficant improvements, while our third approach shows an improvement of up to 20%.

Acknowledgements

This work could not be done without the help and support of my very kind friends and mentors who were always there to help and to support me. Words can not describe the gratitude I have for them. Nevertheless, this section serves to thank the special people who helped me complete my thesis.

First of all, I want to show my great respect and thanks to professor Jimmy Huang, for his encouragement and passionate academic support. Professor Huang's kindness and support constantly motivates me to work hard, and achieve my academic goals.

Furthermore, I would like to thank professor Yang, also in my supervisory committee, for her help and support. Finally, I would like to thank my external examiner, professor Song, for her time and consideration.

I also want to thank Dr. Mariam Daoud for her help and support. None of this would be possible without her guidance. I would also like to greatly thank my friend Karl Aeen for his continued support and guidance.

Apart from the academic support, I would like to thank my friends and family who were there for me through thick and thin. My best friend, and a younger sister at heart, Atanaz, was one of the reasons for my successes over the past few years. Her family was also very helpful in providing me with an easy transition for my new life in Canada.

Finally, I want to thank the most important people in my life; these people gave me all they had, emotionaly and financially, through love and kindness to help me get to where I am today: my Lovely parents, Zari Pirjedi and Reza Forghani Raissi; and my beautiful sister, Mona. I am and where I am today because of my family's love.

To Zari, Reza and Mona

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Background

Information Retrieval is the act of retrieving specific information that is required from a set of resources, or a collection of documents. In other words, it is the act of separating relevant information from irrelevant information in a collection of data. A great example of this in action is Google's search engine. Through this service, people are able to use search terms, or queries[1], in order to retrieve the relevant information they are looking for. Google's data collection is massive, and the whole point of the search engine is to separate relevant data from irrelevant data. This is information retrieval at its heart.

In the field of Information Retrieval, there are many great challenges to be solved. In order to solve these challenges, many organizations exist that aid in this process. TREC, which stands for Text REtrieval Conference (TREC, http://trec.nist.gov), is a major

[1] A query is one or more words that are used as cues to find the relevant information from a dataset.

Conference [2] that provides standardization for researchers in the field Information Retrieval. It provides complete datasets and benchmark information in order to help Information Retrieval researchers have a standard baseline for their work. Once a year, TREC releases queries and documents that researchers use in order to perform research. Furthermore TREC provides a Golden Standard that is created by extracting the relevant reports for each topic (query). The judges are chosen from the medical community, or from general participants of the TREC Medical Track who have significant medical knowledge; normally, the chosen participants posses a PhD in the life sciences [13].

One of the main motivations for the existence of TREC is the collection of documents that it provides to researchers. A "collection of documents" is the raw data that researchers use to conduct novel methods of Information Retrieval on. TREC creates the standard for the collection of documents and the performance baselines. In the early days of Information Retrieval research, document collections were typically very small. During the 1990s, collections were only as big as a few megabytes, or as much information that is present in a 20-page essay [36]. Thus, it did not represent real world databases of document collections. The second problem was that the sharing of such collection of documents was very difficult; researchers were not able to efficiently share their data amongst themselves. Furthermore, most of the companies had their own databases, but they were not really interested to share them with other researchers. Hence, TREC was designed to help all industrial, academic and government researchers to work and

[2] TREC is part of the US National Institute of Standards and Technology (NIST, http://www.nist.gov/)

compete together efficiently in a standardized system. This way, researchers can correctly compare the accuracy and speed of their Information Retrieval algorithms. TREC defines itself not only as an annual competition, but also as a way to share ideas and techniques for performing successful Information Retrieval [22]. The ultimate goal of TREC is to advance the field of Information Retrieval.

1.2 Motivation

Information Retrieval systems generally function by taking as input a query, or a set of queries, and then produce as output the part of the data that is most relevant to the input query given earlier. Medical Information Retrieval is subset of Information Retrieval that is optimized and designed to be used specifically with medical data. Medical literature can be one of the hardest texts to perform information retrieval on, and this is because of issues like *synonyms* and *very-long keyword*s that are common in the medical field. Considering these problems, medical text retrieval is tricky. Yet, it is one of the most essential and critical parts of Information Retrieval. Medical text retrieval is extremely important because it has the potential to reduce the spread of diseases, and ultimately save lives. Furthermore, it is important in order to recognize health-related information that doctors and medical experts need, sometimes very urgently. Therefore, Medical Information Retrieval requires not only impeccable accuracy, but also a very high-speed performance. Medical text information retrieval can be broken down into three major steps: a) identifying information sources, b) using those sources to retrieve relevant information and, in the end, c) analyzing, understanding and using the information in order to deliver the correct diagnoses [13]. In short, the purpose of medical text information retrieval is to help make the decision of which treatment or which medicine to prescribe; this decision can be made by the patient, the patient's family, or the health care professional.

There are various different kinds of factors that can effect health-related decisions. Most important is that medical decisions be based on evidences and facts. The field of medical information retrieval hopes to discover those evidences and facts.

1.2.1 E-Health Initiatives

In Canada, e-health has become a huge movement that is tasked with converting almost all medical records and data to an electronic format. The 2012 Canada Health Info way (https://www.infoway-inforoute.ca/) commissioned study found that the use of Electronic Medical Records helped save taxpayers $177 million in the year 2012 alone [31]. There are many more initiatives to store more health records in an electronic format. Every province in Canada has at least one or more initiatives to bring ambulatory, pharmaceutical, and general health records to an electronic format. All this wealth of information will require state of the art IT infrastructure in order to handle the data. Amongst the IT requirements, Medical Information Retrieval Systems will have a very high priority. As more and more information is collected, medical professionals will need to sift through massive amounts of data in order to find the relevant information. This is a daunting task, and the field of Medical Information Retrieval has never been so important in the history of Canada.

There are also several initiatives around the world that aim to advance the medical field by transitioning it to the electronic age. The GSMA mHealth Program (www.gsma.com) reports that the global eHealth market value is around $160 billion and a growth rate of up to 16% [21]. This shows the need for better and improved Healthcare IT systems in the near Future.

1.3 Problem Definition

My work relies on the subset of the TREC dataset that is related to the Medical field[3], also known as the TREC Medical Data Track[4]. The main goal of my research is to improve the relevancy of medical data retrieval from a set of documents based on a *medical query* given by the user. The medical query can consist of one or more words[5].

The problem that is faced today in the field of Medical Information Retrieval is the low accuracy of search results. This is mainly due to the very specific and unusual requirements of medical terms; in general, medical search terms are very complicated to

[3] The TREK datasets are divided into different "tracks". The particular track I will be working with is the Medical Records Track. The goal of the Medical Data Track is *"to foster research on providing content-based access to the free-text fields of electronic medical records."*

[4] http://trec.nist.gov/tracks.html

[5] In this context, the queries, or "words" need not be actual English words appearing in the dictionary. Any number of letters grouped together can form a single "word".

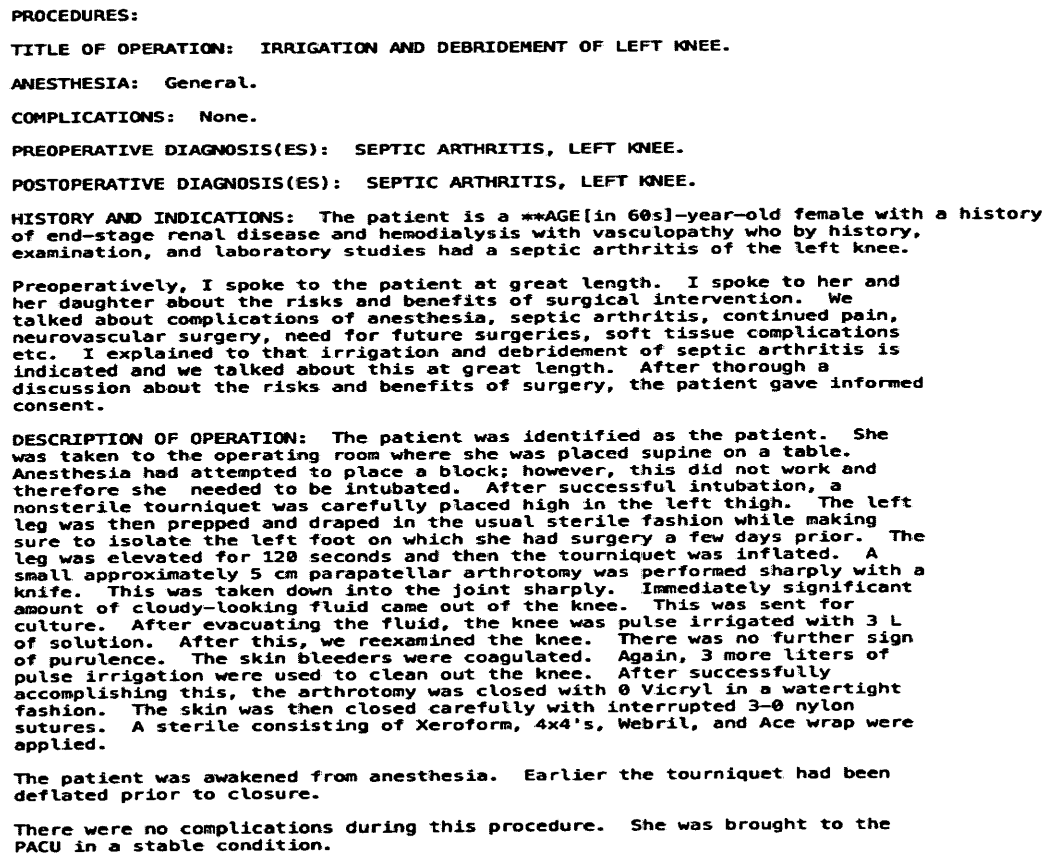
be processed by regular information retrieval methods. As such, the field of Medical Information Retrieval aims to design information retrieval solutions specific to the medical field.

1.3.1 Query Model

The query model in Medical Information Retrieval is very unique. Unlike regular search queries, we do not need to match exact query terms (i.e. Google). For instance, searching "York University" in Google will get you results that match "york", "university", or both "york" and "university". This usually is the ideal situation for non-medical data sets. However, in medical datasets, the queries might not need to directly match the data results. For instance, if the medical query is "heart disease", then the results must include not only results that match "heart" and "disease", but also results that include actual types of heart diseases (i.e. Cardiovascular disease, Cardiac dysrhythmias). Furthermore, perhaps there might be a need to include related diseases as well. Finally, there might also be a need to include other diseases or illnesses that might contribute to heart diseases (i.e. atherosclerosis, hypertension). These use cases show that regular searching algorithms are not sufficient for the field of Medical Information Retrieval.

1.3.2 Medical Dataset Model

Another unique characteristic of Medical Information Retrieval is the design of the medical dataset. Generally speaking, and in the case of the TREC medical dataset, the medical dataset is divided into hundreds of thousands of patient visit records. These records are not grouped together for a single patient; a patient can have multiple records for each of the multiple visits to the medical professional. These files are usually in the XML[6] format, which makes the job of the searching algorithm a little easier. Figure 1 shows a sample report from the TREC Medical Track dataset. The XML tags in this sample report include PROCEDURE, COMPLICATIONS, and DESCRIPTION OF OPERATION.



```
PROCEDURES:

TITLE OF OPERATION:  IRRIGATION AND DEBRIDEMENT OF LEFT KNEE.

ANESTHESIA:  General.

COMPLICATIONS:  None.

PREOPERATIVE DIAGNOSIS(ES):  SEPTIC ARTHRITIS, LEFT KNEE.

POSTOPERATIVE DIAGNOSIS(ES):  SEPTIC ARTHRITIS, LEFT KNEE.

HISTORY AND INDICATIONS:  The patient is a **AGE[in 60s]-year-old female with a history
of end-stage renal disease and hemodialysis with vasculopathy who by history,
examination, and laboratory studies had a septic arthritis of the left knee.

Preoperatively, I spoke to the patient at great length.  I spoke to her and
her daughter about the risks and benefits of surgical intervention.  We
talked about complications of anesthesia, septic arthritis, continued pain,
neurovascular surgery, need for future surgeries, soft tissue complications
etc.  I explained to that irrigation and debridement of septic arthritis is
indicated and we talked about this at great length.  After thorough a
discussion about the risks and benefits of surgery, the patient gave informed
consent.

DESCRIPTION OF OPERATION:  The patient was identified as the patient.  She
was taken to the operating room where she was placed supine on a table.
Anesthesia had attempted to place a block; however, this did not work and
therefore she  needed to be intubated.  After successful intubation, a
nonsterile tourniquet was carefully placed high in the left thigh.  The left
leg was then prepped and draped in the usual sterile fashion while making
sure to isolate the left foot on which she had surgery a few days prior.  The
leg was elevated for 120 seconds and then the tourniquet was inflated.  A
small approximately 5 cm parapatellar arthrotomy was performed sharply with a
knife.  This was taken down into the joint sharply.  Immediately significant
amount of cloudy-looking fluid came out of the knee.  This was sent for
culture.  After evacuating the fluid, the knee was pulse irrigated with 3 L
of solution.  After this, we reexamined the knee.  There was no further sign
of purulence.  The skin bleeders were coagulated.  Again, 3 more liters of
pulse irrigation were used to clean out the knee.  After successfully
accomplishing this, the arthrotomy was closed with 0 Vicryl in a watertight
fashion.  The skin was then closed carefully with interrupted 3-0 nylon
sutures.  A sterile consisting of Xeroform, 4x4's, Webril, and Ace wrap were
applied.

The patient was awakened from anesthesia.  Earlier the tourniquet had been
deflated prior to closure.

There were no complications during this procedure.  She was brought to the
PACU in a stable condition.
```

Figure 1: Sample Report

[6] XML stands for Extensible Markup Language and is a computer language that transforms a human readable document format into a computer readable one. This is done by the use of tags to divide the different sections of a document.

Medical TREC challenge also provides Topics or queries. These topics were created by physicians who were also students in the Oregon Health & Science University (OHSU) Biomedical Informatics Graduate Program. Their goal was to develop number of topics that each matched a reasonable number of visits (more than a few but less than several hundred) in the record set [13]. In Medical TREC 2011, they first chose 35 queries and then they decreased to 34. These queries were the best ones which fit to the collection.

1.4 Contribution

The contribution of this thesis is a novel method for medical information retrieval that improves the relevancy of search results by 3% to 20%. My approach adapts existing Information Retrieval concepts and methods to the field of Medical Information Retrieval.

My approach uses a combination of ICD-9 codes, query expansion methods, sub-type filtering, and re-ranking methods in order to find the most relevant results in a large dataset. ICD-9[7] codes are standardized codes for all the different types of diseases and medical terms used by medical professionals (http://icd9cm.chrisendres.com/). Some examples of ICD-9 codes and their descriptions are shown in Table 1.

[7] ICD-9 stands for International Classification of Diseases – 9th Revision

ICD-9 code	Description
10.41	Repair of symblepharon with free graft
20.01	Myringotomy with insertion of tube
27.31	Local excision or destruction of lesion or tissue of bony palate
52.86	Transplantation of cells of Islets of Langerhans, not otherwise specified

Table 1: Sample of ICD-9 codes

1.4.1 Query Expansion

Query Expansion is defined as a method of "reformulating" an existing query, in order to improve the relevancy of the search results. For instance, if a given query is "heart sickness", the query expansion algorithm might reformulate the query into "heart cardiac disease" in the hopes of improving the relevancy of the information retrieval.

In my work, I use query expansion methods by extracting key terms from a set of already retrieved set of documents to reformulate the initial query, and, ultimately, to improve the accuracy of the search results. The set of already retrieved documents is performed by the current baseline information retrieval system (i.e. Terrier[8]).

Sub-type filtering is a method whereby is using concepts with specific subtype of medical concepts. The ones that we considered are the most important ones which are known as disease and procedure. These subtypes are the ones that are came as a question or

[8] Terrier is an open-source search engine that is commonly used in Information Retrieval research settings.

statement in the queries. This kind of filtering can be useful to discard the unnecessary concepts that can cause lower precision in retrieved information.

Another way of using retrieved information in baseline to improve the results is using re-ranking methods which involves applying similarity measures between the results and the query in order to re-rank them based on the new weighting model.

1.4.2 Methods Proposed

In total, I proposed three different methods for improving Medical Information Retrieval. The first method (Chapter 3) is a re-ranking method based on the cosine similarity[9] between the queries and the data in the retrieved medical reports. This method involves applying the Apriori[10] and the FP-Growth[11] algorithms in order to expand the initial given query. Using cosine similarity, and the concepts produced by the Apriori and the FP-growth algorithms, the retrieval process is run again, this time with an expanded query. The process completes with a set of document results that have been re-ranked.

[9] Cosine Similarity is defined at the similarity of two vectors by measuring the cosine of the angle between them. In other words, it measures the similarity of two sets of information based on a single subject matter.

10 The Apriori Algirthm applies association rule mining in a dataset. This algorithm is further explained in section 3.1.2.1.

[11] The FP-Growth algorithm applies association rule mining in a dataset. FP stands for frequent pattern. This algorithm is further explained in section 3.1.2.2.

The second method (Chapter 4) involves using query reformulation with the usage of subtype filtering[12] and the Apriori Algorithm in order to improve the medical information retrieval. This method uses the retrieved documents from the baseline information retrieval system in order to find the subtypes within them. In turn, using the Apriori algorithm, the subtypes are ranked based on the index; the highest ranked subtypes are then used to reformulate the original query.

The third method (Chapter 5) applies ICD-9 codes and query expansion techniques in order to improve relevancy in medical information retrieval. This method applies top-ranked baseline ICD-9 code descriptions in order to expand the queries. The expanded queries are then re-run in order to improve the accuracy of the retrieved documents.

1.5 Findings

For each of the three methods I have performed, the findings are explained below:

1.5.1 Method 1: APFP-Cosine Algorithm

In this method, we found the results to be negative. In the best case, this method resulted in lowering accuracy by 26% to 71%. We believe that the reason for this extreme regression in accuracy is that applying cosine similarity is not a suitable solution for the

[12] Subtype filtering involves locating sub-concepts from the main concepts in the retrieved documents.

case of medical information retrieval. These results show the lack of relationship between related medical concepts.

1.5.2 Method 2: Sub-AP Algorithm

In this method, the results showed improvements in all the evaluation measures. The improvements show an increase in accuracy from 3% to 17%. These results are strong and show the power of applying subtype filtering in the domain of Medical Information Retrieval.

1.5.3 Method 3: ICD9-Top Algorithm

In this method, applying ICD-9 codes description had a high positive impact on our evaluation measures. The result of this method had the highest improvement among the other proposed methods. In this Algorithm, we showed an increase in accuracy from 7% to 20%. These results are very promising and highlight the power of using ICD-9 codes in the domain of Medical Information Retrieval systems.

1.6 Thesis Structure

In the next section, we will the literature review of materials that are related to this research work. We will illustrate the differences of the work in this research project, with those of previous research work. Chapter 3 describes our first proposed method, which is based on using Cosine Similarity measures; we will call this method the APFP-Cosine method. In chapter 4, we will describe our second proposed method, which is based on using Subtypes of the concepts from the retrieved results; we will call this method the Sub-AP method. Our third and final proposed method is explained in Chapter 5. This method is based on the combination of query expansion techniques and ICD-9 codes from the top retrieved results; we will call this method ICD9-Top. Chapter 6 presents our evaluation measures, which is largely based the TREC evaluation measures. The rest of the thesis belongs to the results and the discussion points (Chapter 7 and Chapter 8). Chapter 9 illustrates the conclusion and future work related to this proposed methodologies.

2 Literature Review

2.1 Re-ranking based on Similarity Calculation

In [29], the authors tried to find a new scoring method in order to re-rank the resulting documents from an information retrieval system (i.e. search engine). This new scoring method involved mapping each retrieved report to a single patient visit. Based on the number of retrieved reports that point to a single visit, that visit is scored accordingly. The higher the number of reports for a single visit means a higher score for that visit in this re-ranking.

Another interesting step that the researchers took was to increase the size of the index. This is a pre-computation step that is done in hopes of bettering the results of the retrieved documents. The researchers expanded the index database by using ICD-9 codes in order to better understand the medical records, and increase the knowledge base of the index.

2.2 Query Expansion

Over the recent decades, the volume of medical literature has increased dramatically. With the general goal of improving medical information retrieval methods, there are several different types of search engines that are designed for medical text retrieval; each search engine uses a different method in order to improve the retrieval accuracy. Apart from the general limitations and challenges of information retrieval, in Medical Information Retrieval, the problem is further compounded by the fact that many users of medical information retrieval systems might not have enough domain knowledge. Therefore, it becomes increasingly difficult to understand the user's query, and provide the results he/she is actually looking for.

This particular problem of "lack of domain knowledge" has been partially solved by a technique called Query Expansion. This technique was proposed by the authors in [42], [20], [40], [46], [7], [27] and [23]. Query Expansion is the process of re-creating the seed query in order to improve the information retrieval results. The process of re-creating the seed query usually involves adding terms to the query, based on the initial retrieved results from the initial seed query. Using Query Expansion, the authors in [38], have shown an improvement of 20-30%. This is an immense improvement that is seldom seen in Information Retrieval research nowadays.

Query expansion becomes especially important when there is a word mismatch. This problem can happen because of the possibility of using different words to define the same concept. When a query contains a commonly used word, then the query will match to more documents, and possibly, to incorrect one due to the word mismatch problem. As Xu has stated, this word mismatch problem becomes very serious when the queries are shorter and spontaneous; this is true since shorter queries have a higher chance of containing terms that co-occurring in a larger set of documents [16]. Nowadays, because of the Internet, shorter queries are more common than longer queries (i.e. Casual users using Google). Mismatch has been a big problem in the Information Retrieval field for a long time.

2.2.1 Automatic Query Expansion

There are two major types of Query Expansion techniques: automatic and manual [27]. Automatic techniques have major advantages compared to manual ones such as relevance feedback and manual thesauri because there is no need for user efforts. Automatic Query Expansion techniques can be divided in two categories, global and local [16].

Global Automatic Query Expansion techniques need corpus-wide statistics, which can take a large amount of computer resources (i.e. time, memory, energy). For instance, we would need the co-occurrence data values for all possible pairs of terms in a collection. If the collection contains ten thousand documents (a very small collection by today's standards), then there are about 3,124,998,800,000 total pairs of words[13]. If we were to

[13] Assuming 250 words per page.

store just one byte of information for each pair of words, then we would need at least 12 terabytes of storage[14].

On the other hand, Local Automatic Query Expansion techniques only process a small amount of top-ranked documents which are retrieved for that specific query. A local technique may use some global statistics such as the document frequency of a term, which should be cheap to store and calculate. In this technique, the source of expansion terms is the set of top-ranked documents. In other words, for Local Automatic Query Expansion Techniques, we first run the regular Information Retrieval system, then use the top ranked results as a starting point for the query expansion techniques.

The simplest local technique is local feedback, which assumes the top-ranked documents are relevant and uses the standard relevance feedback for query expansion procedures.

The relevance feedback process, which was first introduced in the middle of the 1960s, is a controlled automatic process for query reformulation (i.e. query expansion) [14]. There is another similar technique, which was proposed earlier in [41]. Here, the information from the top-ranked documents is used for the estimation of the probability of query terms in the relevant set for that specific query. These terms usually are the ones with the most frequency, excluding stop words from the top ranked documents. Stop words are the words that are filtered out before or after processing of natural language data (text). Stop

[14] Assuming 4.5 characters per word.

words can cause problems in searching the phrases including them. Table 2 shows a partial list of Stop Words.

Lately, research shows that local feedback has a generally good impact on retrieval improvements. However, some have shown that there is a chance to degrade the retrieval performance if the top-ranked documents are not relevant to the query [20]. In this case, a small problem in the retrieval process is compounded due to the assumption that the initial set of retrieved documents is truly the most relevant documents.

The idea of using the initial top-ranked documents for query retrieval improvement was first proposed in 1977. In [35], term clustering had been used on the top-ranked retrieved documents, and the results were used for query expansion. Term Clustering is a simple process of grouping together the most commonly appearing terms in a set of documents.

a	it	these
about	its	they
again	itself	this
all	just	those
almost	few	through
also	from	thus
although	made	to
always	mainly	upon
among	make	use
an	may	used
and	mg	using
another	might	various
any	m	very
are	my	was
as	most	we
at	mostly	were

Table 2: Partial List of Stop Words

There are many more techniques of applying Automatic Query Expansion Techniques. In [32], three different strategies are proposed for Automatic Query Expansion: Synonym-based, Topic model-based and Predication based. In the first method (Synonym-based), the authors attempted to use a few selected UMLS source vocabularies and included lexical variants in the expanded queries. The UMLS is Unified Medical Language System and includes a set of files and software that brings together many health and medical vocabularies and standards into one database, in order to enable interoperability among computer systems. In the second method, Topic model-based, the authors added related terms based on the topic-model trained on 100,000 clinical documents. The third and final method is the Predication-based expansion technique that uses a large predication database, extracted from global medical literature by a natural language processing (NLP) system called SemRep. The authors in [32] showed that all three methods resulted in improvements of up to 23% in comparison with baseline.

In [6], another method for applying query expansion techniques on data from the TREC Medical Track dataset had been proposed. The authors suggested using two external sources: the UMLS database and the Cengage Learning's collection of medical reference encyclopedias. The research showed an improvement of 6% in some of the evaluation measures.

Furthermore, the authors in [18] proposed two new approaches for query expansion. The first approach is the Default Query Expansion, which uses terms from the top ranked documents. In this method, the authors collected the terms with the most frequency from the top 10 documents; in this case, 10 terms are added to each query. The second approach is called Concept Extraction; in this method, the authors annotated the queries with concepts from the UMLS database using the MetaMap system. MetaMap is a widely available program that can provide access to the concepts in the UMLS Metathesaurus[15] medical text. MetaMap was founded in order to improve medical text retrieval, especially for the retrieval of MEDLINE[16]/PubMed[17] citations [1].

The authors in [18] used the MetaMap concept list and their short descriptions to perform query expansion. Their results showed that the second approach had slight improvements in precision. However, the performance decreased in the general case, when compared to the default query expansion technique.

2.2.2 Manual Query Expansion

In [37] researches tried both manual and automatic query expansion techniques in conjunction with the addition of ICD-9 codes, based on the original queries using the

[15] Metathesaurus is a National Cancer Institute browser containing different biomedical vocabularies, including the International Classification of Diseases for Oncology.

[16] MEDLINE (Medical Literature Analysis and Retrieval System Online) is a bibliographic database of life sciences and biomedical information.

[17] PubMed is a free database accessing primarily the MEDLINE database of references and abstracts on life sciences and biomedical topics.

MeSH[18] and MetaMap databases. In the results, the manual run had good results but the automatic run did not perform particularly very well.

In [38] the authors attempted to use ICD-9 codes in order to expand the queries. The authors collected the ICD-9 code definitions and their relationships from the UMLS database. Next, the authors added the newly learned definitions into the original queries. For the experiments, the authors ran three separate runs based on three different parts of the documents[19]. These three different runs are based on the different parts of the documents that had been used to execute the expanded queries on them. These methods lead them to get improvements in their runs in compare with baseline.

2.2.3 Association Rule Mining Algorithms

Association rule mining algorithms have been used within query expansion techniques in many different ways. Association rule mining consists of two phases: Rule Generation and Frequent Itemset Discovery. The Rule Generation involved using the mined frequent itemsets in order to generate rules. This step is trivial and takes very little time, only 1-2% of the time of the entire information retrieval process. Thus, association rule mining algorithms focus on finding frequent itemsets. A frequent itemset is defined as the itemset which has a support that is higher than the user-specified support.

[18] Medical Subject Headings (MeSH) is a comprehensive controlled vocabulary for the purpose of indexing journal articles and books in the life sciences; it can also serve as a thesaurus that facilitates searching.

[19] In this case, documents refer to the documents that were used for the query expansion step.

In the field of Information Retrieval, the two most popular frequent itemset mining techniques are Eclat and Apriori. Eclat is a depth-first search using set intersection at its heart. In Eclat, for each item, we store a list of transaction IDs, or TIDs, in a vertical layout. The advantage of the Eclat algorithm is that it has very fast support counting. However, the disadvantage is that intermediary TIDs can become too massive to hold in memory [30]. The Apriori Algorithm [34], on the other hand, uses a breadth first approach and uses data structures such as a Hash Tree in order to efficiently count itemsets. Apriori's "bottom-up" approach and pruning techniques make it the most viable and popular choice for frequent itemset mining. Researchers also propose the DHP [19], the PARTITION [1] and the CD algorithms [33]. There are Many more algorithms that researchers have proposed or used in their work [3]. The best characteristics of Apriori Algorithm are its simplicity and superiority of performance and also the fact that it is scalable with large data sets.

Another powerful and popular algorithm that is well-known in the data mining field is FP-growth. The FP-Growth algorithm is another algorithm that is used to find itemsets, but without using Candidate Generations. This Candidate Generations technique is a “bottom-up” approach used by the Apriori Algorithm whereby each term is considered as a candidate to one of the available item sets. It has been shown that applying the Candidate Generations technique is very time consuming and memory-consuming, and hence, the FP-growth algorithm has a much higher performance.

2.3 Weighting Models

In query expansion methods, there are different types of weighting models that can be used. In [45] the authors discovered an alternative technique to utilize query expansion by attaching weights to query terms based on the term's distribution among the various categories. The authors presented the normalized entropy (NE) method to determine the special category for each term. They derived two supervised weighting schemes. The authors used the TREC dataset and it was shown that the schemes which are included in the traditional IDF have significant outperform for queries which containing more than a few specific terms and also has the competitive results on short and well-defined queries.

3 APFP-Cosine Method

The TREC Medical track dataset contains a corpus about 101 thousand anonymous medical records from the University of Pittsburgh NLP[20] Repository. The collection consists of one month of structured reports from multiple hospitals and includes nine types of reports from different departments in those hospitals. The medical report types include Radiology reports, History and Physicals reports, Consultation reports, Emergency Department reports, Progress Note reports, Discharge Summary reports, Operative reports, Surgical Pathology reports, and Cardiology reports. Each of these medical reports can be mapped to one of 17,265 patient visits. A patient visit is an individual stay at hospital by a patient. Each report belongs to a single patient, however, many reports can represent a single patient [29].

In this research, we developed a new methodology based on Association Rule Mining and cosine similarity measures. We wanted to find similarity measures between assigned

[20] Natural Language Processing

queries and medical reports in order improve on the retrieval accuracy, compared to the baseline results.

Using Apriori and FP-Growth algorithms in combination with cosine similarity are the foundations of APFP-Cosine methodology.

The entire APFP-Cosine Re-ranking process is shown graphically and in detail in Figure 2.

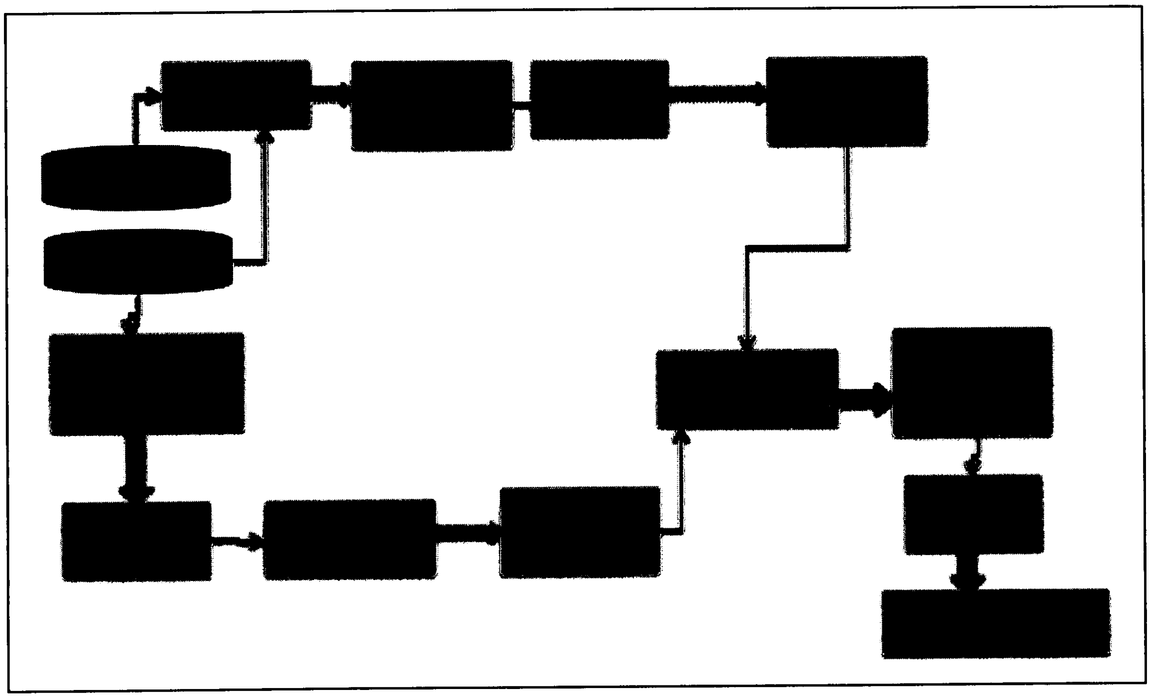


Figure 2: Re-ranking Method Diagram

3.1 APFP-Cosine Methodology Steps

3.1.1 Indexing Based on Using UMLS and BioLabeler

As we mentioned earlier, the initial queries contain the sentences and words that describe the medical conditions or treatments or even disease and procedures that a user is searching for. In order to proceed to next step, we first need to perform conceptual indexing[21] for the queries and the reports. This resulting index is based on using UMLS and an online biomedical text mining tool named BioLabeler[22], which associates UMLS concepts to the data in any given text [25].

The produced index file contains rows and columns of data that will be used for performing our weighting algorithms. The rows represent each of the many concepts found in each report. The columns represent extra information for each of the concepts in the rows. The columns show information in this order: report ID, concept, number of concepts in that specific report, number of reports that includes that specific concept, number of that concept in the whole collection, and also type of the concept which could be a disorders or a procedure or anatomy. This index file is required to provide the different parameters that are needed for the weight calculations for each of the concepts in the next step.

[21] Conceptual Indexing is the act of indexing on a collection of documents, based on a set of concepts.

[22] http://www.biolabeler.com/bioLabeler/

3.1.2 Using Association Rule Mining for Query Expansion

Association rule mining is a well-known method for finding the relationship among different variables in very big data collection. The goal is to find strong relationships between terms in a data collection. Association rule mining is used in various applications, including web mining, intrusion detection, continuous production and bioinformatics.

3.1.2.1 Apriori Algorithm

The Apriori algorithm is one of the most effective algorithms for dividing a large data collection into smaller sets of related terms. These sets of closely related terms are called frequent item sets. The Apriori algorithm is used to look, or to mine, for the frequent item sets of boolean association rules[23]. This algorithm can be divided into two sub-algorithms: First, retrieve all item sets that have a support[24] which is greater than the minimum; this will be called the frequent item. And second, using the previously found frequent item set, we will generate all the association rules. At this point, for each item set X, all non-empty subsets of A are found; in other words, if *support(A)/support(a)* $\geq$ *minimum confidence*, then the association rule is A-a. In other words, we would like to exploit the association rules that have a high confidence, specifically a confidence level that is not below the user specified level [24].

[23] Boolean Association rules says that each item in the dataset is considered to be either part of an itemset, or not part of at all.

[24] Support is defined as the total number of reports that include specific concept divided by total number of reports

3.1.2.2 FP-Growth

The FP-Growth algorithm is another algorithm that is used to find itemsets, but without using the Candidate Generations. Candidate Generations is a "bottom-up" technique used by the Apriori algorithm whereby each term is considered as a candidate to one of the available item sets. In turns out that applying the Candidate Generations technique is very time consuming, and hence, the FP-growth algorithm has a much better performance. The FP-Growth algorithm is a tree-based approach that uses a divide-and-conquer strategy. Behind the scenes, the FP-Growth algorithm uses a special data structure named the Frequent Pattern Tree, or the FP-Tree, which preserves the itemset association data. The FP-Tree is a data structure that provides quick Specifically, the FP-Growth algorithm works as such:

1. Reduce the collection database to represent frequent itemsets, with the usage of an FP-tree data structure.
2. Next, we divide the reduced database into several conditional databases; each database represents a single frequent pattern.
3. Finally, we mine each database individually.

By following this algorithm, the FP-Growth algorithm reduces the costs of searching for patterns. This is done by recursively searching for shorter patterns, then concatenating them into the longer patterns. The tree-approach of the FP-Growth algorithm allows it to use memory efficiently, while also providing a quick response time when looking for patterns. This provides good selectivity.

3.1.2.3 Query Expansion

The query expansion step is possible after the execution of the Apriori and the FP-Growth algorithms. The top 100 reports (along with its associated concepts) from the baseline retrieval method are given to the algorithms. The concepts are divided into semantic groups. The semantic groups we used for these runs are DISO (Disorders), PROC (Procedures), PHYS (Physiology), CHEM (Chemical and Drugs) and ANAT (Anatomy). In the results, we disregarded general concepts such as CONC (Concepts and Ideas), GEOG (Geographic Areas) and LIVB (Living Beings).

The Apriori and FP-Growth algorithms produce the weights for each of the concepts. We only considered concepts which had a support ≥ 0.6. Our experiments confirmed that the optimal value for the support is 0.6. These selected concepts are then added to the original concepts of the initial query, thereby forming a new, expanded query.

3.1.3 Vectors of Queries

The 34 given queries[25], which are the only clues to finding the relevant information in the large corpus, are analyzed in order to find the UMLS concepts. These queries are then defined into groups of concepts, based on their weights, which is derived from their support values. The definition of Support is provided in Equation 1.

[25] The 34 given queries are the queries provided by TREC that are to be used for research purposes.

$$Sup_C = N_t/N$$

Sup_c = Support value for concept C

Nt = Total number of reports that include C

N = Total number of reports in collection

Equation 1: Support Formula

Presenting concepts with their respective support values, allows us to create a vector of concepts that is associated with each query term. This vector can be defined as shown in Equation 2:

$$V_Q = [C_1,S_1 \ \ C_2,S_2 \ \ C_3,S_3 \ldots C_i,S_i]$$

C = Concept
S = Support

Equation 2: Query Vector

3.1.4 TF.IDF Scoring

TF.IDF, or Term Frequency – Inverse Document Frequency is a scoring method which is indicates how relevant single term is in relation to the entire document, or corpus. For our method, the TF.IDF scores can be obtained through the index file, which was produced in the first few steps by Terrier. For this step, we use the TF.IDF method to produce a score for each concept in each of the medical reports; using this information, a vector of concepts, along with their TF.IDF values is generated for each medical report in the entire corpus. Calculation of the TF.IDF is shown on Equation 3.

$$(TF.IDF)_{Concept} = TF \times IDF = TF \times \log\frac{N}{N_t}$$

TF= Concept Frequency in the specific report

N= Total Numbers of reports= 17011

N_t= Total number of reports that contained this specific concept

Equation 3: TF.IDF for each concept of each report

3.1.5 Vectors of Reports

All the values that have been mentioned in 3.1.4 can be generated by using data from the indexed file. The result of this calculation leads us to a vector, which contains all the concepts and their weights. Each Report's vector can be defined as shown Equation 4.

$$V_{Report} = [C_1, TF.IDF_1 \;\; C_2, TF.IDF_2 \;\; C_3, TF.IDF_3 \ldots C_i, TF.IDF_i]$$

C = Concept
TF.IDF = *from* Equation 3

Equation 4: Report Vector

3.1.6 Cosine Similarity

The last step of this re-ranking method is to find and to calculate the re-ranking score for each report [26].

We calculate the Similarity values for each pair of vectors (VQUERY, VREPORT).

[26] Here, each report signifies the reports that were the results of running our baseline approach (i.e. Terrier).

There are several different types of similarity measurements in vector space model[27].

In the vector space model, we will need to use similarity measurements that are based on the inner product of the vectors. There are several types of vector similarity measurements, such as Jaccard, Dice and Cosine [10].

The Jaccard Similarity Coefficient, also know as Tanimoto Similarity is a similarity ratio given over bitmaps of a fixed size vector. This is ratio is basically the number of common bits, divided by the number of bits set in the either sample. In our research, as we mentioned in 3.1.3 and 3.1.5, the vectors of our reports and our queries are not of a fixed size nature. Further, since our vectors contain weight measurements of the terms, it is not possible to use this similarity measurement; the Jaccard model accepts either 0 or 1 inside the vectors. Therefore, this measurement model was discarded as a possible choice for our similarity measurements.

The Dice similarity measurement is also known as Dice's coefficient or Sorenson index. For two vectors of keywords, the dice coefficient is defined as twice the shared information over the sum of cardinalities of the vectors. Therefore this measure is not suitable for our purposes since the cardinality measurements of vectors are something we are not considering.

The Cosine Similarity measurement is the most popular choice for vector similarity measurements in Information Retrieval. In the vector space model, we can simply use the angles as the measure of divergence between the vectors. Then, in order to have a

[27] In the vector space model, text is represented by a vector of terms.

numerical similarity, the angle value is converted by applying the cosine calculation. This way, we end up with a numerical value for the similarity between two vectors. This method is especially useful in Information Retrieval since identical vectors receive a similarity measurement of 1, and orthogonal vectors receive similarity measurement of 0. As an alternative, we can also use the dot-product (inner-product). However, the problem with the dot-product calculation is that it takes into account the length of the vector, something that we want to avoid in Information Retrieval [39].

Finding Cosine Similarity values is a result of calculations of the two-mentioned vectors in section 3.1.3 and 3.1.5; this is shown in Equation 5.

$$\text{Score (Report)} = \text{Cosine } (V_Q, V_{\text{Report}}) = \frac{\sum Sup(C_i) \times TF.IDF(C_i\ ,\ Report_n)}{\|V_Q\| \times \|Doc_n\|}$$

Equation 5: Cosine Similarity

3.1.7 Re-ranking based on Cosine Similarity score:

The final step in this APFP-Cosine method is to re-rank the reports based on the cosine similarity values generated in the previous step. The Cosine Similarity values score are calculated for all the reports in the entire corpus. After completing the above-mentioned steps, we then feed in the re-ranked reports back into Terrier, in order to perform retrieval evaluation measurements. This step is further described in section 6.

3.2 Algorithm

```
1 List collection = Trec.data();
2 Array queries = initialQueries();
3 collection.performUMLS.Biolabeler();
4 queries.performUMLS.BioLabeler();
5 List collectionIndexed = collection.performIndex();

6 Array expandedQueryAP = queries.performApriori();
7 Array expandedQueryFP = queries.performFPGrowth();

8 Array reportVector = RV.vector(collectionIndexed);

9  // Below block for FPGrowth method
10 Array queryVectorFP = QV.vector(expandedQueryFP);
11 // Below called CosineSimilarity Class
12 List rankedReportsFP = CS.rank(queryVectorFP,reportVector);
13 // results conatins the output of running Terrier
14 List resultsFP = Terrier.evaluate(rankedResportsFP);

15 // Below block for Apriori Method
16 Array queryVectorAP = QV.vector(expandedQueryAP);
17 // Below called CosineSimilarity Class
18 List rankedReportsAP = CS.rank(queryVectorAP,reportVector);
19 // results conatins the output of running Terrier
20 List resultsAP = Terrier.evaluate(rankedResponseAP);
```

Figure 3: Re-ranking based on the Cosine Similarity Algorithm

Figure 3 shows the detailed algorithm for the APFP-Cosine re-ranking method. In the first 5 lines, the collection and the query are indexed based on UMLS concepts. The next two lines perform the Apriori and the FP-Growth algorithms on the queries, in order to perform query expansion for this method. Line 8 extracts the report vector from the database index. Since we are performing the re-ranking and the cosine similarity measures separately, lines 9-14 represent the steps for the FP-Growth Algorithm, while lines 15-20 represent the steps for the Apriori Algorithm; in each of these blocks, we first extract the expanded query vector, then rank the reports, based on this newly formulated query. The results are then stored in a variable of type List (line 14 and line 20).

4 Sub-AP Method

The TREC Medical Track data records contain various different types of medical terms and concepts. Finding the most valuable ones is a challenge. Solving this challenge will lead us to correctly respond the users' queries.

In this method, two of the most important subtypes (Disorders and Procedures) are selected and filtered from the indexed collection. These two subtypes were chosen because these were the most popular subtypes amongst the queries. This matter is further discussed in 4.1.2. After gathering the desirable concepts' subtypes, the Apriori algorithm is applied in order to get the weight of these concepts in order to expand the queries and retrieve the results.

The summary of this method is shown in Figure 4.

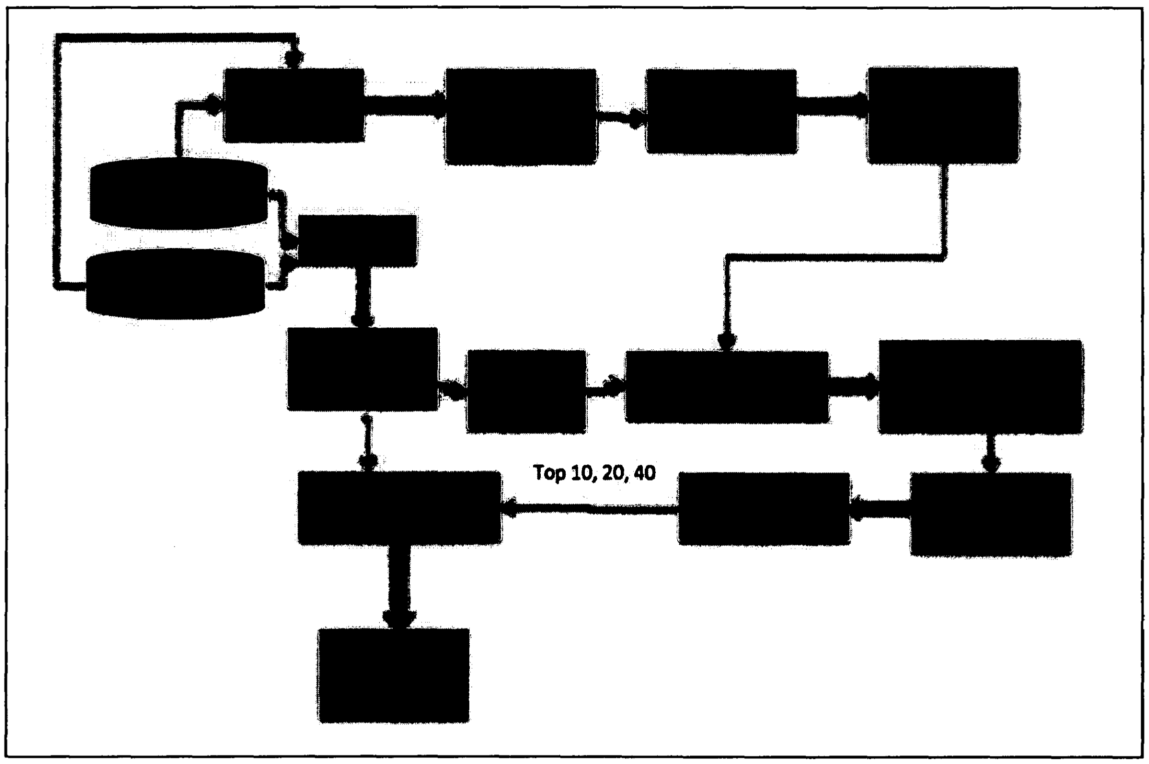


Figure 4: Subtypes Method's Diagram

4.1 Sub-AP Methodology Steps

4.1.1 Indexed Collection

As we mentioned in section 3, the collection consist of just over17000 visits. A *Visit* is defined as all the reports for all the visits for a single patient. In this collection, more than one report can belong to the same patient.

As mentioned in chapter 3, the initial stage involves creating two files: 1) an index of the entire corpus and 2) the baseline results from an information retrieval system, such as

Terrier. These 2 files can be based either on the reports or on the Visits, which contain these reports. In the final stages of this method, the evaluation (Golden Standard) is done via the reports, and not the Visits. Therefore, we will need to map our results (Visits) to that which is suitable for our evaluation (reports). The mapping file for determining which reports belongs to which Visits is available from TREC.

4.1.2 Subtypes Filtering

This step involves filtering the subtypes (PROC, DISO) from the initial index. The index file contains all the visits' concepts, along with the specification of their types and their visit IDs. The subtypes are shown in Table 3.

ACTI	Activities & Behaviors
ANAT	Anatomy
CHEM	Chemicals & Drugs
CONC	Concepts & Ideas
DEVI	Devices
DISO	Disorders
GENE	Genes & Molecular Sequences
GEOG	Geographic Areas
LIVB	Living Beings
OBJC	Objects
OCCU	Occupations
ORGA	Organizations
PHEN	Phenomena
PHYS	Physiology
PROC	Procedures

Table 3: Different Tags of Subtypes

Based on the medical definitions of the queries, we are able to locate the subtypes that best match the queries. The most important subtypes for us are the concepts that belong to the Procedures and to the Disorders (Diseases) subtypes.

We use a java program (A.4) to filter the desired subtypes from the entire indexed file. This new smaller indexed file contains all the concepts from the visits that are under the DISO and PROC subtypes.

4.1.3 Using Top Ranked Results

We used the top-ranked baseline retrieval results in order to find the desirable concepts. For the purposes of this thesis project, we simply used the top 50 results when looking for the desirable concepts. It is also possible to use a different number of top-ranked results. The reason why the top 50 was chosen is because this number of visits is not so large to give us irrelevant concepts. Furthermore, it is also not that small, which might cause it not to give us enough number of concepts.

4.1.4 Finding Most Relevant Concepts

This next step is finding the associated concepts with these top relevant visits. A java program was used to find the associated concepts with the top ranked visits that was gathered from the previous steps.

4.1.5 Apriori Algorithm Scores

The Apriori algorithm can calculate the weight of the subtype-related concepts. This output consists of each concept and the produced weight calculated by the Apriori algorithm. This combination of (concept, weight) will be used for the next step.

4.1.6 Re-ranking and Combining Scores

Perl is very suitable for the re-ranking step, based on the weighted queries. The prepared query with the concepts and the associated weights from 4.1.5 is run through a Perl program. In order to generate new score for each document and apply the re-ranking step, we used a score combination technique whereby combine the new score of the new query and the old baseline score. This score is calculated as show in Equation 6.

$$\text{NewScore(Doc\}}=\propto\times\text{Score(Doc)}+(1-\propto)\times\text{BaselineScore(Doc)}$$

Equation 6: Score Combination

4.2 Algorithm

As previously mentioned, this method is based on using the subtype-related concepts as a filter in choosing the specific relative concepts; these concepts are later used in creating an expanded query. The step-by-step algorithm for this method is shown in Figure 5.

In the first 5 lines, the collection and the query are indexed based on UMLS concepts. The next three lines run the Terrier baseline algorithm and extract its top 50 ranked documents. Next, we retrieve the two subtypes we are interested in, namely, DISO (disorder) and PROC (procedure). In line 10, we use the extracted subtypes in order to filter the subtype-related concepts from the top 50 baseline-ranked documents. Next, we perform Apriori on the concepts we gathered in the previous step. Here, we expand the initial query in order to get a newly formulated, larger query. This query is then used (line 13) to get the results for this Sub-AP method. Line 12 calculates the score for the baseline method, while line 13 calculates the score for this new Sub-AP method.

```
1 List collection = Trec.data();
2 Array queries = initialQueries();
3 collection.performUMLS.Biolabeler();
4 queries.performUMLS.BioLabeler();
5 List collectionIndexed = collection.performIndex();

// Run Terrier and extract top 50 results
6 Terrier.initialize(collection, queries);
7 List resultTerrier = Terrier.run();
8 List resTop50 = resultTerrier.getTop(50);

// Find the two subtypes we are working with
9 List subTypeCollect = collectionIndexed.SubType("DISO,PROC");

// Extract the subtypes from the TOP 50 results
10 Array TopSubTypes = subTypeCollect.filter(resTop50);

// Perform Apriori on the Top subtypes
11 Array expandedQueryFP = TopSubTypes.performApriori();

// Below, we have the two results: baseline, and our new method
12 List resultsBaseline = Perl.runScore(expandedQueryFP);
13 List resultsExpanded = Perl.runScore(queries);

// Below contains the evaluation results
14 Array evalRes = Evaluate(resultsBaseline, resultsExpanded);
```

Figure 5: Sub-AP Algorithm

5 ICD9-Top

Using ICD-9 codes in order to expand medical-related queries has been used in different ways by many researchers. The main challenge is trying to find the most relevant ICD-9 codes in relation to the query terms. Only this way we can get the best possible retrieval performance. In this method we attempted to apply different query-expansion methodologies by using the most relevant ICD-9 codes. For this method, we gathered ICD-9 codes from the Top 20 and Top 50 retrieved results from baseline.

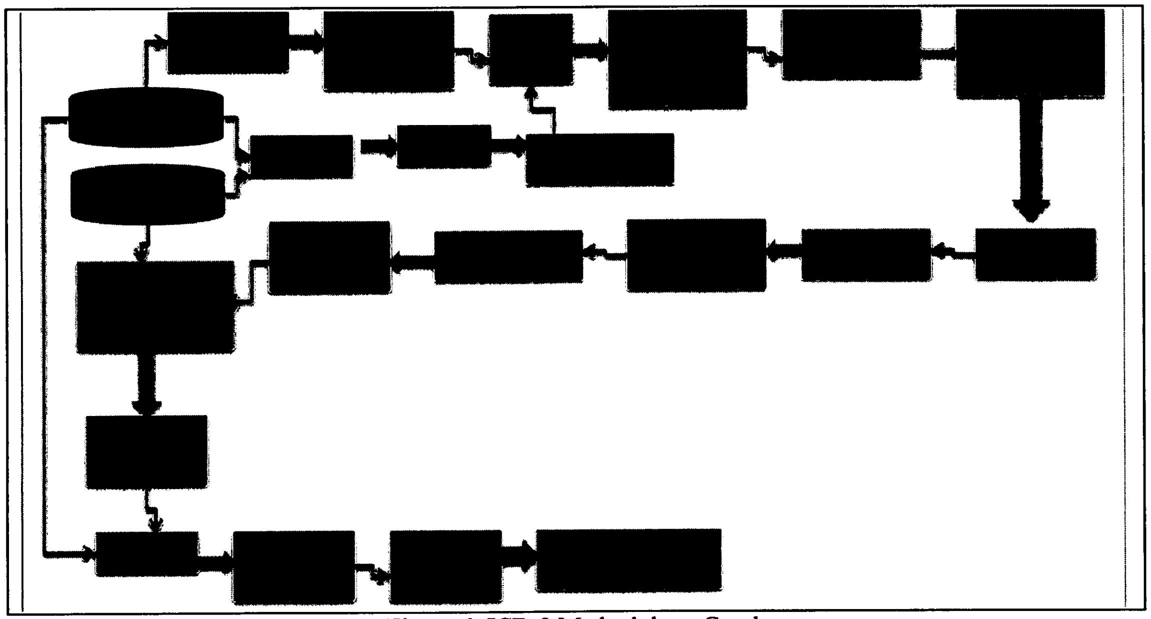
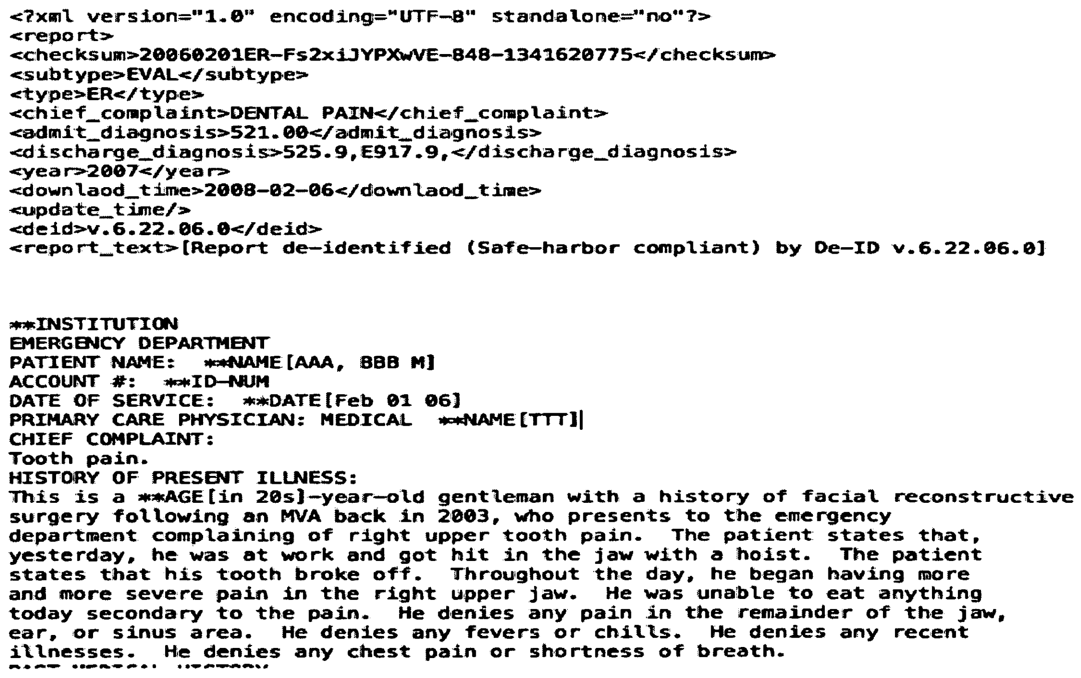


Figure 6: ICD-9 Methodology Graph

5.1 Methodology Steps in Details

The TREC Medical Track data collection is a massive collection of patient records available to researchers. It is made up of actual meetings between doctors and patients. The entire medical corpus is made up of 100,000 documents collected from doctors' reports. These reports are gathered from the almost 17,000 number of patient visits. Each report is formatted in the XML format[28]. Figure 7 shows an example of a report in the TREC Medical Track data collection. In the top part of the report, there are XML tags (meta-data) such as: *chief_complaint*, *admit_diagnosis*, and *discharge_diagnosis*. In-between these tags, descriptions or numbers (actual-data) can be found.



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<report>
<checksum>20060201ER-Fs2xiJYPXwVE-848-1341620775</checksum>
<subtype>EVAL</subtype>
<type>ER</type>
<chief_complaint>DENTAL PAIN</chief_complaint>
<admit_diagnosis>521.00</admit_diagnosis>
<discharge_diagnosis>525.9,E917.9,</discharge_diagnosis>
<year>2007</year>
<downlaod_time>2008-02-06</downlaod_time>
<update_time/>
<deid>v.6.22.06.0</deid>
<report_text>[Report de-identified (Safe-harbor compliant) by De-ID v.6.22.06.0]


**INSTITUTION
EMERGENCY DEPARTMENT
PATIENT NAME:  **NAME[AAA, BBB M]
ACCOUNT #:  **ID-NUM
DATE OF SERVICE:  **DATE[Feb 01 06]
PRIMARY CARE PHYSICIAN: MEDICAL  **NAME[TTT]|
CHIEF COMPLAINT:
Tooth pain.
HISTORY OF PRESENT ILLNESS:
This is a **AGE[in 20s]-year-old gentleman with a history of facial reconstructive
surgery following an MVA back in 2003, who presents to the emergency
department complaining of right upper tooth pain.  The patient states that,
yesterday, he was at work and got hit in the jaw with a hoist.  The patient
states that his tooth broke off.  Throughout the day, he began having more
and more severe pain in the right upper jaw.  He was unable to eat anything
today secondary to the pain.  He denies any pain in the remainder of the jaw,
ear, or sinus area.  He denies any fevers or chills.  He denies any recent
illnesses.  He denies any chest pain or shortness of breath.
```

Figure 7: Sample Report along with the tags

[28] The XML format is a simple text formatting technique that divides each section of the text into a specific category. (Diagnosis, medication, etc.)

In order to use ICD-9 codes, we gathered all the codes with their detailed descriptions[29]. At this point, we had the ICD-9 codes and their descriptions ready to be used against the medical TREC collection. A Java program was written in order to read each report and find both of the desired tags (Admit Diagnosis, Discharge Diagnosis), and put them in an output. These two tags' definitions are as follows: Admit Diagnosis includes information on what diseases the patients were admitted into the hospital for. Discharge Diagnosis includes diagnosis for the patients at the time of discharge. These tags are the only two tags at the top of each report that have the specific code belonging to the diseases and the procedures performed for each patient at the hospital or clinic. For this reason, why chose these two tags as our desired tags for this method.

5.1.1 Baseline

The input to Terrier[30] is the entire Corpus of reports from our medical TREC collection. Terrier is an open source search engine that is used in order to index and retrieve collections. Terrier is written in Java and is easy to use. Using vanilla Terrier (unmodified) provides baseline retrieval results for the initial queries. The input is the TREC medical collection that consists of almost 100,000 reports that are entirely in the XML format, and a query file, which contains 34 queries.

[29] ICD-9 Codes and their descriptions were gathered from http://www.icd9data.com/

[30] Source files and documentations for Terrier can be found at http://terrier.org/

After Terrier has completed a run, an output file is created that contains the results of the search, which we will now call the baseline benchmark results. Terrier uses the optimal document weighting model and the retrieval approach; Terrier is regarded as the state of the art retrieval system [15]. The output file produced contains the top one thousand results for each of the 34 queries[31]. The reports are ordered in descending level of relevance, with the top item to be the most relevant result.

Apart from searching, another usable function of Terrier is Indexing. Indexing is the process of creating a mapping between key-words and their frequencies in a collection. The purpose of indexing is to improve the performance of querying, or retrieving data in a collection. For the TREC collection, we used Terrier to create the indexed file. The various components of the Terrier diagram can be seen in Figure 8.

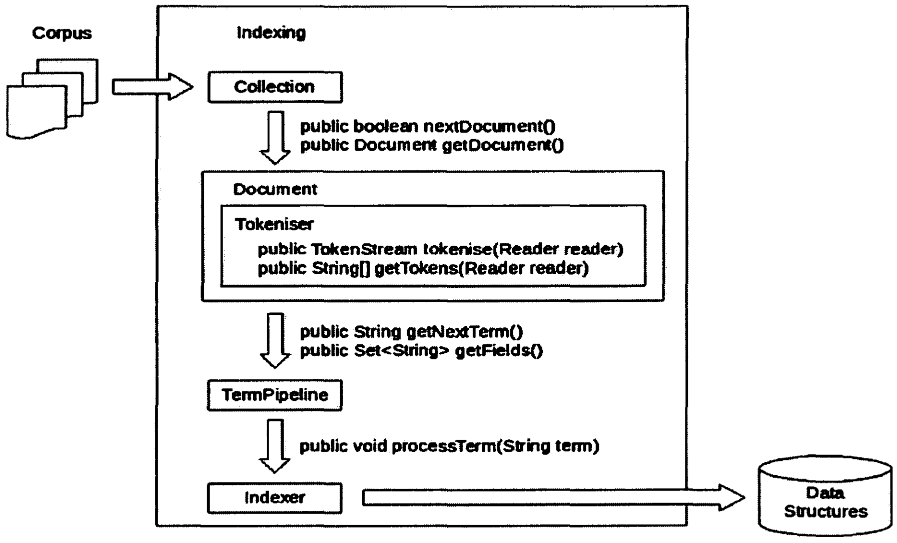

Corpus
Indexing
Collection
public boolean nextDocument()
public Document getDocument()
Document
Tokeniser
public TokenStream tokenise(Reader reader)
public String[] getTokens(Reader reader)
public String getNextTerm()
public Set<String> getFields()
TermPipeline
public void processTerm(String term)
Indexer
Data Structures

Figure 8: Terrier Components

[31] There are a total of 34,000 reports in the output file after running Terrier with 34 queries.

5.1.2 Gathering ICD-9 Codes

In this step, we go through all the reports (documents) and read the specific tags that we need in order to find the desired ICD-9 codes. For this method, these tags included: *Admit Diagnosis* and *Discharge Diagnosis*. These two tags are exhibited through various different ICD-9 codes that are present in the reports. For example, in one of the reports, the *admit diagnosis* tag includes: *521.00*, which, according to the ICD-9 code description, is: *Diseases of hard tissues of teeth Dental caries unspecified.* Analyzing the report text further, we learn that the patient was a man who was admitted with upper tooth pain that led to a piece of tooth breaking off at his place of work.

For the other target tag, *Discharge Diagnosis,* the codes are *525.9* and *E917.9.* The first code is described as *Other diseases and conditions of the teeth and supporting structures and unspecified disorder of the teeth and supporting structures* and the second code is described as *Multiple gestation placenta status and other striking against with or without subsequent fall.* Each of these tags can be used as clues to the other parts of the text in the document. A Java program was written in order to read all the reports and find the target tags' ICD-9 codes, and separate them into a file (A.6). This output includes each report's check sum, which is the report ID, and the ICD-9 codes it contain

5.1.3 Gathering Top Documents

The next step is to find the best matched documents. Finding the best results needs a set of pre-defined settings. To get the best possible results, we considered the top 5, 10, 20, 50, and 100 ranked documents from the Terrier baseline for each query. We did this because these top documents are the most relative documents to use for the query expansion steps. This step concludes by organizing the top files into separate groups that will be used for the next step.

5.1.4 Finding Associated ICD-9 codes

Each of the top X[32] reports had been written to an output[33] with their associated ICD-9 codes. In this step, we look up the ICD-9 codes for each of the top X documents from the output of step 5.1.2. These ICD-9 codes will be used in next step.

5.1.5 Ranking ICD-9 Codes

In this step, we gathered all top X reports with their associated ICD-9 codes for each separate query. For example, the query could be "101" with its top 20 reports. For these 20 reports, we have already gathered all its associated ICD-9 codes from step 5.1.4. Each of the ICD-9 codes are further organized and sorted by their frequencies, or the number of times they are associated with each report. For instance, the ICD-9 code "*389.9*" has a

[32] (X= 5, 10, 20, 50, or 100).

[33] The output is further described in section 6.4.2.

frequency of 9 in the entire top 20 reports. This is the most frequent code for this query in the top 20 relevant reports.

5.1.6 Finding The Top ICD-9 Codes

At this point, we need to choose which amount of the top ranked ICD-9 codes from the top X documents to use for the query expansion step. We must choose the most occurring ICD-9 codes because these are the most relevant ICD-9 codes for each of the queries. This task is difficult, as we need to decide the breadth of the top ICD-9 codes to use. Initially, we considered many different amounts. For example, we could only use the top-most occurring ICD-9 code, but that is not going to help us improve the results. We can, on the other hand, choose all the occurring ICD-9 codes, but that is also not efficient, as some of the ICD-9 codes appear very infrequently, sometimes only once in the top X documents. Therefore, we must find a balance; to choose only the most occurring ICD-9 codes that are highly relevant to the query, and that will actually help us improve the results of the searching algorithm. For this step, we graphed the frequencies of the ICD-9 codes, in order to get a better picture of what threshold to use when deciding which amount of the top ICD-9 codes to use.

Figure 9 shows the ICD-9 codes with their frequencies for a sample query (101) in the Top 20 relevant documents. This figure illustrates the codes behavior in terms of their frequencies. The first most-repeatable code has a frequency of 9 and the all the others have lower numbers. It is observed that the frequency numbers, in descending order, are 9, 5, 4, 4, 3, 3, 3, 3, and 3. This order shows that from fourth code onwards, the frequency *3* is repeated 5 times. Since we are attempting to add the most relevant ICD-9 codes, we would like to select only the documents with the highest occurring ICD-9 codes. Hence, we would like to ignore the pattern of repeated low numbers, as observe in the above example with the repeated 3s.

Figure 10 shows ICD-9 codes ranked based on their frequency for top 50 relevant documents for query (101). As we can assume, the frequency numbers increase, and this time it starts from 13. We observe the same pattern as in Figure 9 for the top 20 related documents repeated. Looking at these two figures simultaneously, we get closer to deciding which number of frequency level to choose as the cut-off in the top ranked documents. In almost all the ranked codes, for all the queries in the top documents, we observe a similar pattern.

After gathering and ranking the ICD-9 codes for the top X[34] documents, a common characteristic was observed in the rankings; the Top 3 ICD-9 codes were much more occurring that the fourth, fifth, and so on. This behavior is shown in the Figure 9 and in Figure 10. Hence, the threshold was chosen to be the top 3 occurring ICD-9 codes; this number will be used in the next step. In effect, by choosing the top 3 most frequent

[34] (X= 5, 10, 20, 50, or 100).

numbers, their associated ICD-9 codes and descriptions will be used in the query expansion steps.

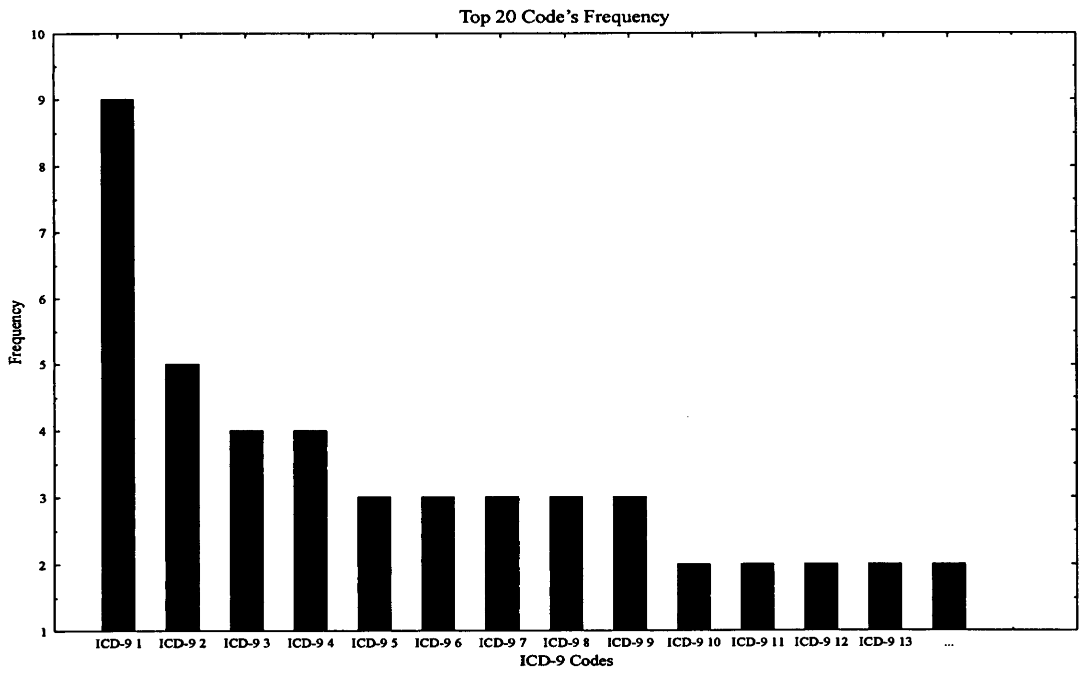


Figure 9: ICD-9 Codes Frequency for Top 20 Relevant reports

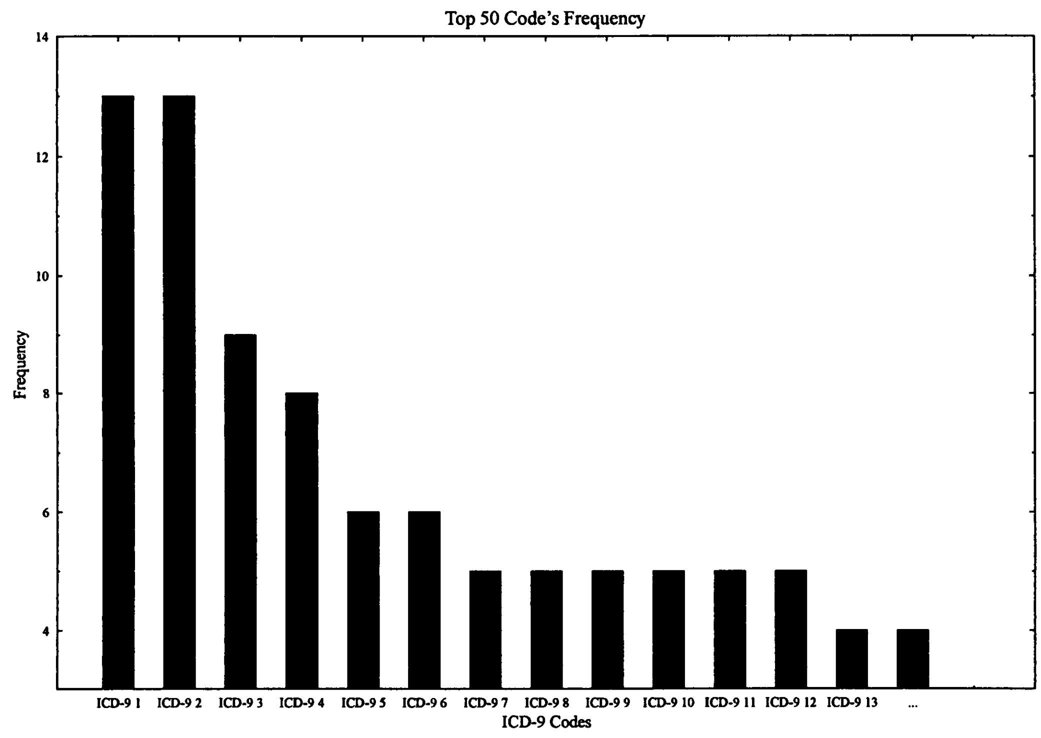


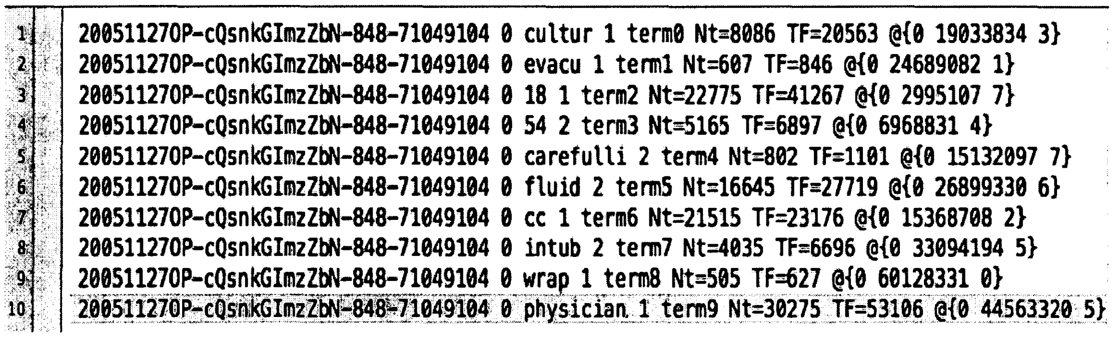
Figure 10: ICD-9 Codes Frequency for Top 50 Relevant reports

5.1.7 ICD-9 Codes Descriptions

In order to gather the descriptions for the ICD-9 codes chosen in the previous step, we referred to the descriptions that we were able to gather online. The descriptions required some changes in order to be suitable for our experiments. The original formatting for the ICD-9 codes were very specific; the format is generally of the nature XX.YY, where XX is the disease number, and YY is the sub-category for that disease. For example, the ICD-9 code *85* is described as *Operations on the breast* and the code *85.12* is described as *pen biopsy of breast*, which refers to a type of *Operations on the breast.* We reformatted the descriptions of the ICD-9 codes to appear correctly for our data processing. The last step is to find the definition of the top 3 ICD-9 codes into different outputs.

5.1.8 Weighting Terms

As described in section 5.1.1, the reports had been indexed by the baseline operation on Terrier. As previously mentioned, the index of a file is defined as a text file that contains the mappings between keywords and their occurrences in a collection. Each line contains different columns. The first column is the unique report ID (indicated by *check_sum*); the second column is the current term being analyzed; the third column is the frequency of that term in the report identified in column one; the fourth column contains the number of reports that include this term (indicated by *Nt*), and the fifth column contains the frequency of the term in the entire collection (indicated by *tf*). An example of this index is shown in graph Figure 11.



```
200511270P-cQsnkGImzZbN-848-71049104 0 cultur 1 term0 Nt=8086 TF=20563 @{0 19033834 3}
200511270P-cQsnkGImzZbN-848-71049104 0 evacu 1 term1 Nt=607 TF=846 @{0 24689082 1}
200511270P-cQsnkGImzZbN-848-71049104 0 18 1 term2 Nt=22775 TF=41267 @{0 2995107 7}
200511270P-cQsnkGImzZbN-848-71049104 0 54 2 term3 Nt=5165 TF=6897 @{0 6968831 4}
200511270P-cQsnkGImzZbN-848-71049104 0 carefulli 2 term4 Nt=802 TF=1101 @{0 15132097 7}
200511270P-cQsnkGImzZbN-848-71049104 0 fluid 2 term5 Nt=16645 TF=27719 @{0 26899330 6}
200511270P-cQsnkGImzZbN-848-71049104 0 cc 1 term6 Nt=21515 TF=23176 @{0 15368708 2}
200511270P-cQsnkGImzZbN-848-71049104 0 intub 2 term7 Nt=4035 TF=6696 @{0 33094194 5}
200511270P-cQsnkGImzZbN-848-71049104 0 wrap 1 term8 Nt=505 TF=627 @{0 60128331 0}
200511270P-cQsnkGImzZbN-848-71049104 0 physician 1 term9 Nt=30275 TF=53106 @{0 44563320 5}
```

Figure 11: Sample Lines of Index

Apart from the actual index, the Terrier indexing process also has some important features that we used. For instance, when feeding a data collection into Terrier, the text files are stemmed. Stemming is the process of removing the prefixes and the suffixes from the words; essentially, stemming returns only the *stem* of a word in the English vocabulary. An example of a Python program that performs stemming is shown in Appendix A.9. Furthermore, Terrier also runs the text through a "stop-word" removal step. Stop-word removal is the process of removing highly occurring words in a document. These highly occurring words are unnecessary for the text-retrieval process, and are therefore useless. An example of a Python program that performs stop-word removal is shown in Appendix A.9.

Having all the terms related to the top-3 ICD-9 codes', as well as their definitions stemmed and stop-word removed, led us to go through the indexed file and finding the associated weights to each term. For this method, we used the TF.IDF weighting model and, as described earlier, all the requirements had been gathered. In Equation 7, the TF.IDF calculation formula and the parameters used is shown:

$$(TF.IDF)_{Term} = TF \times IDF = TF \times \log\frac{N + 0.5}{N_t + 0.5}$$

TF= Term Frequency in the specific report

N= Total Numbers of reports= 17011

N_t= Total number of reports that contained this specific term

Equation 7: TF.IDF score

The weight for each term can be calculated through this formula with the information provided by the indexed file. Since the TF.IDF is calculated for each term in each document, we would need to calculate the average of the TF.IDF score of a term for each query. This calculation is show in Equation 8:

$$weight\ of\ each\ term\ for\ each\ query = \frac{\sum_{i=1}^{X}(TF.IDF)_{term}}{X}$$

Equation 8: Weight of each term for each query

After calculating the average TF.IDF score for each term in each query, we are ready to use these values as weights in order to perform the query expansion in the next step.

5.1.9 Query Expansion

Query expansion is the procedure of reformulating a query in order to improve the retrieval performance. In order to perform the query expansion step, we assigned a weight of 1 to each term from the initial query. Next, we assigned a weight, as calculated by Equation 8 to each additional term from top 3 ICD-9 codes.

5.1.10 Retrieval

In this step, we perform the actual information retrieval step. This step involves using Terrier to retrieve the results and evaluate the results. These settings will describe in chapter 6.

5.2 Algorithm

There are 12 stages in order to complete this algorithm. Below, each step in the algorithm is described further.

```
1 List collection = Trec.data();
2 Array queries = initialQueries();
3 List reportsICD = RR(collection);
4 List indexFile = collection.performIndex();

// Terrier Baseline results
5 List resTerrier = Terrier(queries);
6 List topXDoc = getTopDocs(resTerrier);

// Finding the top ICD from the Baseline
7 List TopDocsICD = FR.findReports(topXDoc, reportsICD);
8 List rankICDFreq = CICD.countICD(TopDocsICD);
9 List top3ICD = getTop3ICD(rankedICDFreq);

// Define and weigh the TOP 3 ICDs
10 List defineTop3ICD = Definition.(top3ICD);
11 List weightsTop3ICD = Weighter.run(defineTop3ICD, indexFile);

// Expand Query, and run Terrier on expanded queries
12 Array expandedQuery = Expander.run(weightsTop3ICD, queries);
13 List resExpandedTerrier = Terrier(expandedQuery);

14 // Below contains the evaluation results
15 Array evalRes = Evaluate(resultsBaseline, resExpandedTerrier);
```

Figure 12: ICD9-Top Algorithm

In the first 5 lines, the collection and the query are indexed based and the reports are read. Line 4 searches the collection for ICD-9 codes. Lines 5 and 6 run the Terrier baseline algorithm and extract its top X[35] ranked documents. Next, in lines 7-9, we find the top ICD-9 codes in the documents. Line 9 extracts only the 3 most occurring ICD-9 code, as described in section 5.1.5 Next, in lines 10 and 11, we define and weigh the top 3 ICD-9

[35] (X= 5, 10, 20, 50, or 100).

codes, in order to perform the query expansion. In lines 12 and 13, we expand the queries, based on the ICD-9 codes, and run Terrier with this newer and expanded query. In line 15, we present the evaluation of our new method.

6 Experimental Settings

6.1 Information Retrieval

6.1.1 Standard Information Retrieval Methods

In the standard IR model, we simply want to compute the similarity, or "closeness", of a query, *q*, to a document, *d*. Figure 13 displays the standard IR algorithm:

```
for each document d in collection
    score calculate similarity of document d to query q
    add above score to score of n top ranking documents
end
```

Figure 13: Standard IR Algorithm

In this procedure, each document is looked at separately, and a score for it is defined based on the query. This method is not optimal, due to its bad performance, and low accuracy in many cases.

6.1.2 Terrier Information Retrieval Method

Terrier is a state-of-the-art Information Retrieval Software that is used in research and in industry. Terrier is an "open-source" software that was created by members of the Information Retrieval Research Group at the University of Glasgow.

The standard model that was originally chosen, as described in 6.1.1, is simply too slow and inaccurate in practice. For this reason, Terrier has since adopted several advanced information retrieval methods.

The inverted method involves using inverted files, we use mappings from query q to its locations in the database, or collection.

6.1.2.1 Inverted Index Retrieval Method

The inverted method involves using inverted files; we use mappings from query q to its locations in the database, or collection.

Figure 14 shows the IR algorithm for inverted indexing:

```
scores array         //Tracks of score for each document
term scores array   //Tracks query term score for each document
for each query term t
        relevant documents =    documents that contain at least
one occurrence of t
          for each document d in array relevant documents
          term score array[d] = score of term t in document d
           end
           for each document d in array relevant documents
          scores array[d] = scores array[d] + term score array[d]
       end
end
scores array = sort(scores array)   //Top documents are highest
                                     ranked documents
```

Figure 14: Inverted Index Algorithm

In the above algorithm, the performance is greatly improved; for this reason, Terrier currently uses this newer approach for ranking documents. One interesting point about Terrier is that it calculates the individual score for each document first. Then, it calculates the total score for all documents; this is done in order to improve performance with regards to the total runtime.

6.2 TF-IDF Retrieval Method

The Term Frequency – Inverse Document Frequency (TF-IDF) model is a retrieval method that attempts to truly reflect how important a term is in a document. The TF-IDF value of a document increases as the number of times the terms increases in the document. However, the TF-IDF decreases proportionally to the total number of times it appears in the entire collection. The calculation for TF.IDF involves two major components: TF and IDF. The calculation for TF involves calculating the number of times a term appears in a document. This calculation is too simple, and will not be demonstrated here. The calculation for IDF is a little more complicated, and is show in Equation 9.

$$IDF = \log(\frac{numberOfDocumentsInCollection}{numberOfDocumentsContainingTerm + 1})$$

Equation 9: IDF formula

The *+1* in the denominator is used in order to avoid calculating *log 0*. As the number of times the term appears in the entire collection increases, the value of IDF decreases for that term in the document. Terrier supports this retrieval method. Here is an important code snippet, taken exactly as it is from the Terrier source code, which calculates the TF-IDF value (Figure 15).

```
public final double score(double tf, double docLength) {
double Robertson_tf = k_1*tf/(tf+k_1*(1
b+b*docLength/averageDocumentLength));
double idf = i.log(numberOfDocuments/documentFrequency+1); return
keyFrequency * Robertson_tf * idf;
}
```

Figure 15: Terrier Source Code to Calculate TF.IDF

Note that Terrier uses a more advanced form of TF-IDF that incorporates two extra constants, *k_1* and *b*, and the document length.

6.3 Perl

Perl is a computer language that was originally used for data processing, especially with text files. Perl is very efficient in dealing with a massive number of, usually large, files. It has recently become very popular, and has grown into a full-fledged general purpose programming language. It is used for both quick one-liner codes, as well as large-scale development. Perl is designed to be efficient, rather than easy to read. Perl combines the look and features of C, java, awk, sed, and sh. One important characteristic of Perl is that it has no inherent limit to the amount of memory it uses. If the system memory allows, it will attempt to insert the whole file (i.e. 3GB) into memory for processing. Furthermore, recursion can be performed in unlimited depths.

Perl was used in various parts of this research project. One example, where Perl was used in order to re-rank the results has been shown in A.5.

6.4 Relevance Judgment

Relevance judgments can be defined as the closeness to *the results that are 100% accurate to the query.* The TREC Medical Track data collection defines Relevance Judgment slightly differently: “If you were writing a report on the subject of the topic and would use the information contained in the document in the report, then the document is relevant”. The judgments are only binary which means only two things: “Relevant” or “Not relevant”. A resulting document cannot be defined as partially relevant. A document is tagged as relevant if any part of it is relevant. In this assumption, there is no condition on the size of the document being judged, in comparison to the entire document volume.

Judging is done using a pooling technique [13] on the set of documents used for the task that year.

Measuring of relevance for the Medical Track dataset was performed manually, as described by the following steps: OHSU[36] hired judges who were physicians and students in the OHSU Biomedical Informatics Graduate Program. 25 physicians judged a number between 1-9 topics, depending on their available time. These judges were instructed to use a rating for each of the visits, in order to determine if such a patient can be a candidate for a clinical study on the relevant topic. A "relevant judgment" means that the patient could be considered as a candidate for the study. A "possibly relevant judgment" means that the patient could be a candidate for the study, but insufficient information was available to make an accurate decision. A "non-relevant judgment" means that the patient cannot a candidate for the clinical study. The judgments have been done for each of the queries by at least one single judge [13].

6.5 Evaluation Parameters

There are several common evaluation measures that can be used for the TREC Medical Track dataset. In this sub-section, we will describe each of them briefly in order to illustrate the application of these measures to our research evaluation.

The effectiveness of Information Retrieval systems is usually measured by *Precision* and by *Recall*. Precision measures the percentage of the returned documents that are actually

[36] Oregon Health & Science University

relevant to the query. Recall measures the percentage of all relevant documents in the collection returned by the system [17].

6.5.1 Recall

Recall is the measure of the ability of a system to present all the relevant items. The standard calculation is shown in **Equation 10**:

$$Recall = \frac{number\ of\ relevant\ items\ retrieved}{number\ of\ relevant\ items\ in\ collection}$$

Equation 10: Recall Formula

6.5.2 Precision

Precision is the measure of the ability of a system to present only the relevant items. The standard calculation is shown in Equation 11:

$$Precision = \frac{number\ of\ relevant\ items\ retrieved}{total\ number\ of\ items\ retrieved}$$

Equation 11: Precision Formula

Precision and Recall are set-based measures. That is, they evaluate the quality of an unordered set of retrieved documents. To evaluate ranked lists, Precision can be plotted against Recall after each retrieved document.

6.5.3 Precision at n

In an information retrieval system that retrieves a ranked list of results, the top-n documents are the first n in the ranking. Precision at n is the proportion of the top-n documents that are relevant.

If r relevant documents have been retrieved, then at rank n, Precision is:

$$\boxed{\textit{Precision at } n = \frac{r}{n}}$$

Equation 12: Precision at n Formula

The value of n can be chosen based on an assumption about how many documents the user will view. In an Internet search, the results page typically contains ten results, so n = 10 is a natural choice. However, not all users will use the scrollbar and look at the full top ten list of results. In a typical setup the user may only look at the first five results before scrolling down, suggesting that a Precision at 5 as a measure of the initial set seen by users. In the our TREC evaluation measures, both Precision at 5 (P5) and Precision at 10 (P10) have been calculated.

It is possible to calculate Precision at a higher cutoff, although Precision will give equal weight to every result in the list. For example, when calculating Precision at 1,000, the 1,000th document is as important as the 1st, whereas users of a ranked retrieval system are likely to consider the 1st document as the most relevant. In order to calculate Precision at n, it is only necessary to obtain relevance judgments for the top-n documents, unlike Recall, which can only be measured if the complete set of relevant documents has been identified [13].

6.5.4 bpref

The bpref measure is designed for situations where relevance judgments are known to be far from complete. It was introduced in the TREC 2005 terabyte track [9]. The bpref computation is designed to have preference for judged relevant documents that are retrieved ahead of judged irrelevant documents. Thus, it is based on the relative ranks of judged documents only. The bpref measure is defined as:

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

Equation 13: bpref Formula

6.5.5 MAP

In recent years, other types of information retrieval measures have become more prevalent. The most common measurement in the TREC community is the Mean Average

Precision (MAP), which calculates a single-figure measure of quality across all recall levels [8]. Among the many evaluation measures, there is good discrimination and stability when using MAP measurements. For a single information need, Average Precision is defined as the average of the precision values achieved from the set of the top "K" documents existing after each relevant document is retrieved; this value is then averaged over the information needs. In other words, if the set of retrieved relevant documents for an information need $q_j \in Q$ is $\left\{d_1, \dots, d_{m_j}\right\}$ and R_{jk} is the set of ranked results from the top results until you get to document d_k, then :

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision\ (R_{jk})$$

Equation 14: MAP Formula

6.5.6 R-precision

For a given query topic Q, R-precision is the precision at R, where R is the number of relevant documents for the query topic Q. In other words, if there are "r" relevant documents among the top-R retrieved documents, then R-precision is:

$$R - precision = \frac{r}{R}$$

Equation 15: R-precision Formula

Hence, R-precision is defined as the proportion of the top-R retrieved documents that are relevant, where R is the number of relevant documents for the current query. This requires pre-ante knowledge of the set of relevant documents. This will be a shallow evaluation measure for a query with a few relevant documents, and a deep evaluation measure for a query with many relevant documents. Rank cutoff R is the point at which Precision and Recall are equal, since at that point both of Recall and Precision are $\frac{r}{R}$.

Average R-precision is the defined as the mean of the R-precision values for an IR system over a set of n query topics. It is expressed as follows:

$$ARP = \frac{1}{n}\sum_{n} RP_n$$

Equation 16: Average R-precision Formula

Where RP represents the R-Precision value for a given topic from the evaluation set of n topics.

7 Results

7.1 APFP-Cosine

This section describes our results for the first method, which is applying the Apriori and the FP-Growth method in order to expand the queries. This is done in conjunction with Cosine Similarities for the re-ranking process.

7.1.1 AP-Cosine Results:

Figure 16 shows the MAP values for the re-ranking method based on the cosine similarity values and the Apriori algorithm. This figures shows all MAP values for each query separately. As shown, there is only one query that has a bigger value than the baseline value, query 109.

R-precision values for this method and their values for each query can be seen in Figure 17, as shown, no query has higher R-precision values than the baseline results. In Figure

18, the bpref values did improve on the baseline values, except with query 106, which showed a +22% improvement; the other queries have lower bpref values compared to the baseline. Therefore, the other queries do not show any improvements.

Figure 19 shows the comparison between the baseline results and application of the Apriori algorithm and the Cosine Similarity method for the P5 Values. The only queries that have improvement are query 101 and 109. P10 values are shown in Figure 20. In this case, there are no improvements in any of the queries when compared to the baseline P10 values.

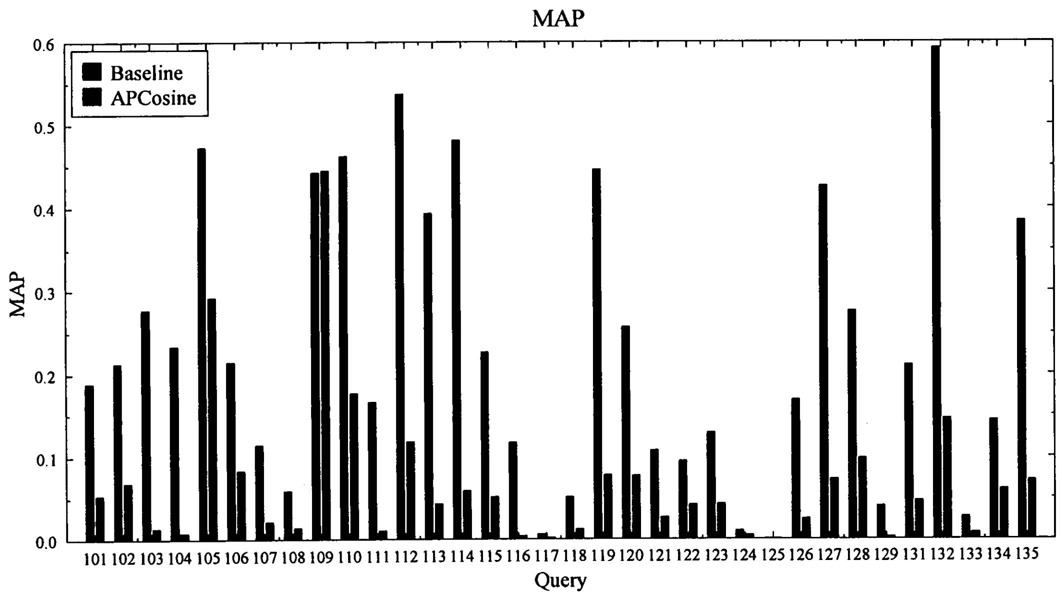


Figure 16: MAP Values for AP-Cosine Method

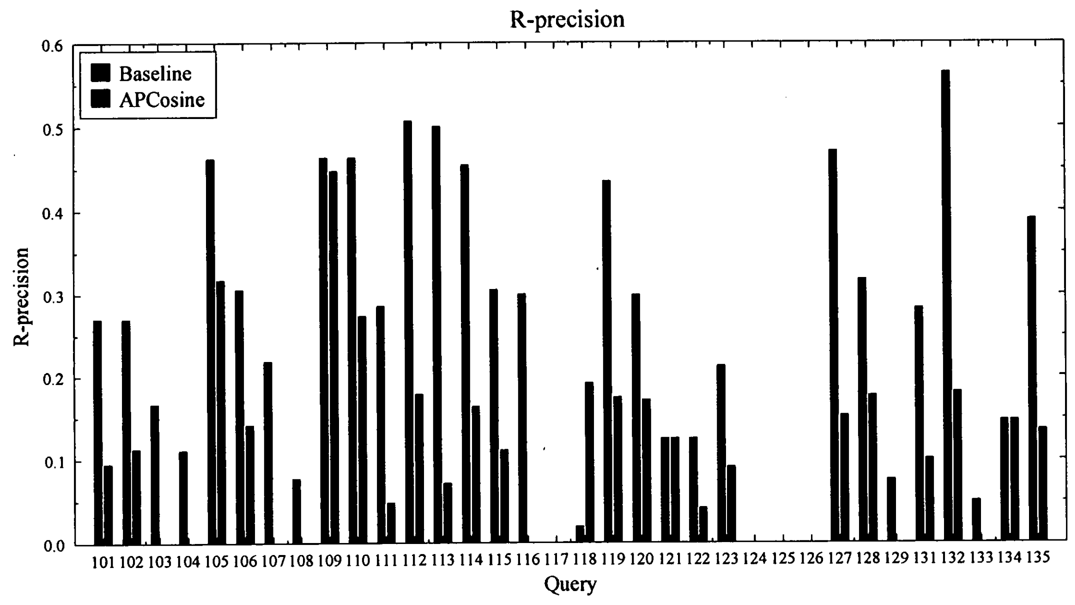


Figure 17: R-precision Values for AP-Cosine Method

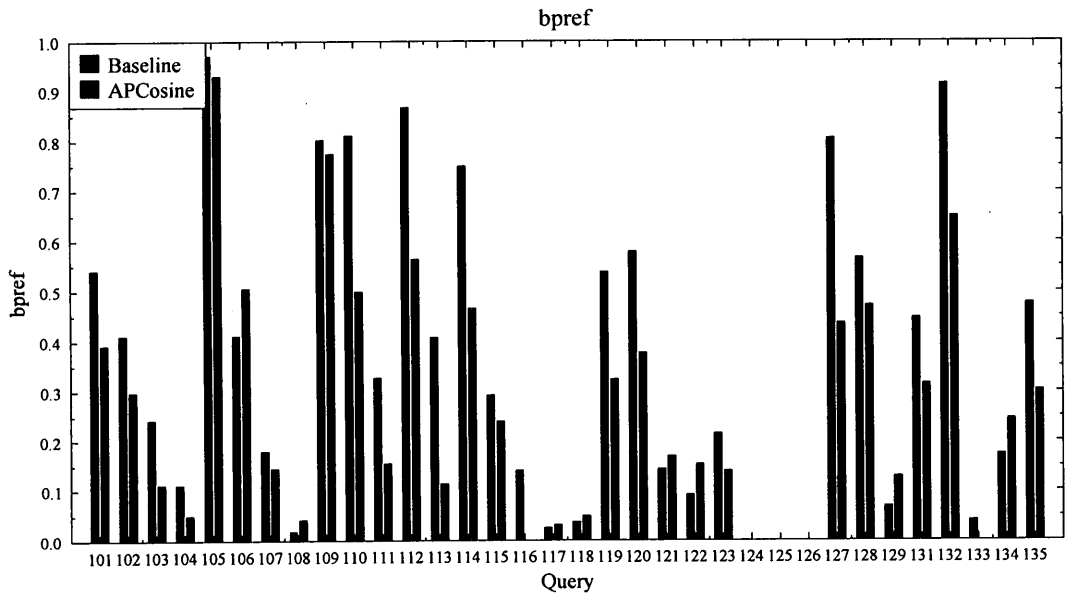


Figure 18: bpref Values for AP-Cosine Method

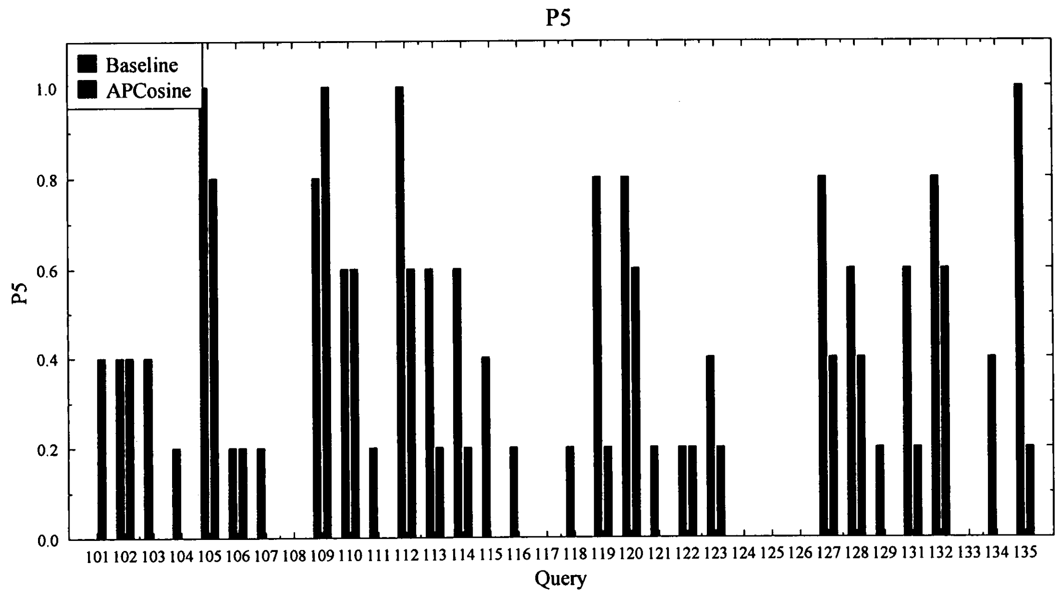


Figure 19: P5 Values for AP-Cosine Method

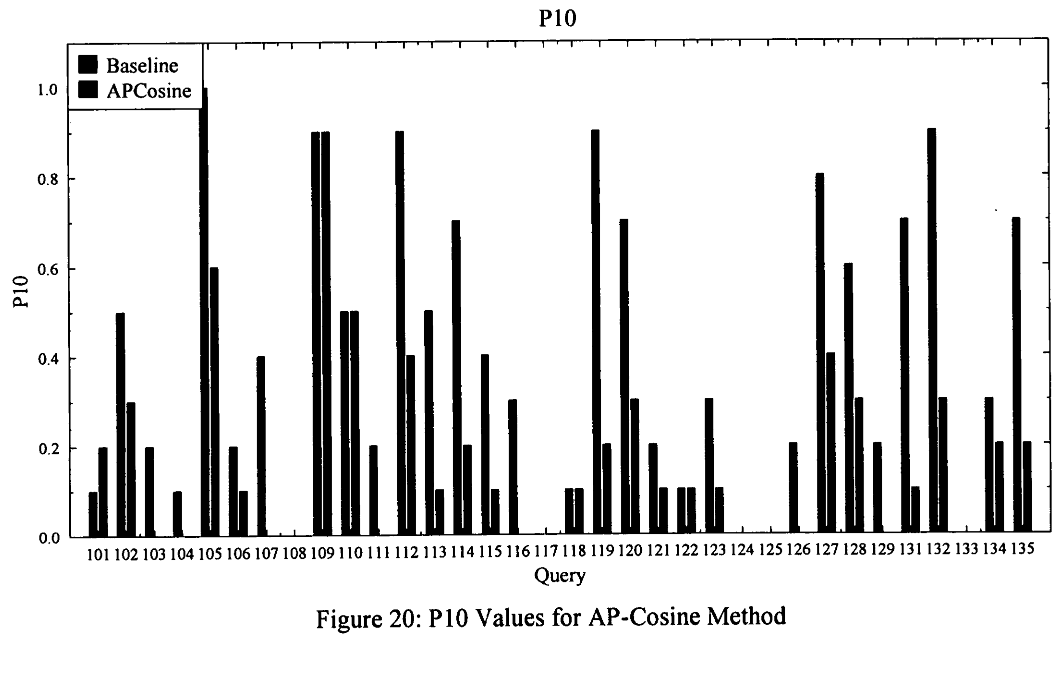


Figure 20: P10 Values for AP-Cosine Method

The summary for this method results with the average of all the evaluation measures is shown in Table 4:

	Map	R-prec	bpref	P5	P10
Baseline	0.2343	0.2551	0.3646	0.4059	0.4000
APCosine	0.0669 (-71%)	0.1022 (-59%)	0.2665 (-26%)	0.2178 (-46%)	0.1706 (-57%)

Table 4: APCosine Method in compare with Baseline

As shown, there are no improvements in any of the evaluation measures. This is due to the ineffectiveness of using cosine similarity in finding specific concepts in the retrieved results. This method shows that, in that case of medical terms, cosine similarity values are not effective in the re-ranking step. It is possible, however, that cosine similarity values are useful if applied in different ways in Medical Information Retrieval.

7.1.2 FP-Cosine Results:

Figure 21 shows the MAP values for each of the queries in the FP-Cosine method. In this figure, the only query that beats the baseline value and has a small improvement is query 109. The R-precision values are shown in Figure 22. R-precision values did not have any improvements in any of the queries. Bpref values are shown in Figure 23. Here, we see that there are no improvements in the any of the queries.

P5 values, in Figure 24, show that the only queries that had improvements are query 101 and query 109. There are three queries that achieve the same P5 values as that of the baseline: query 102, query 106 and query 122.

P10 values rarely changed and showed no signs of improvement in any of the queries. The only query that achieves the same value as that of the baseline is query 109 (Figure 25).

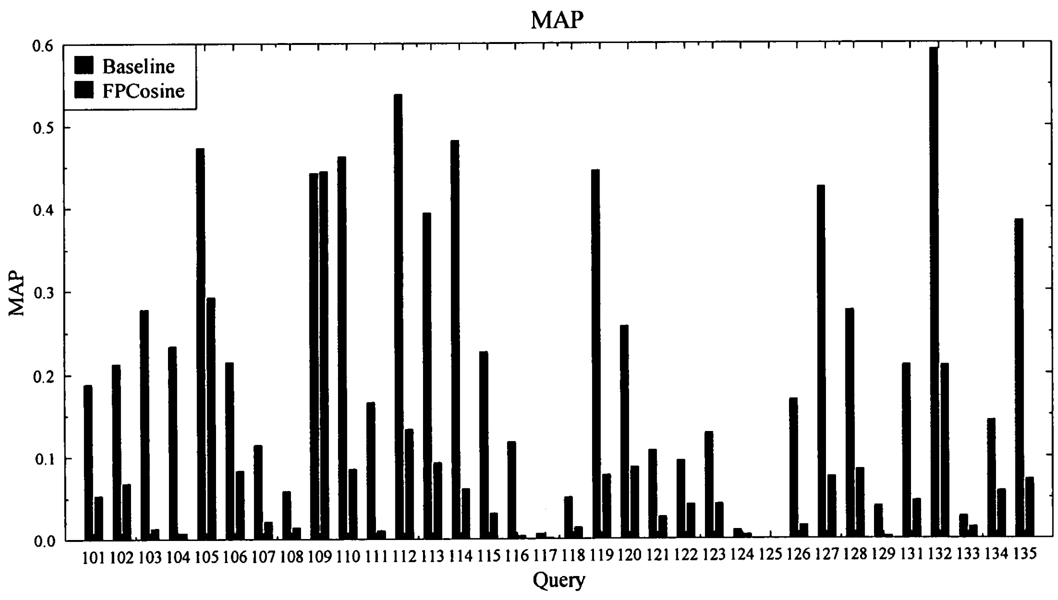


Figure 21: MAP Values for FP-Cosine Method

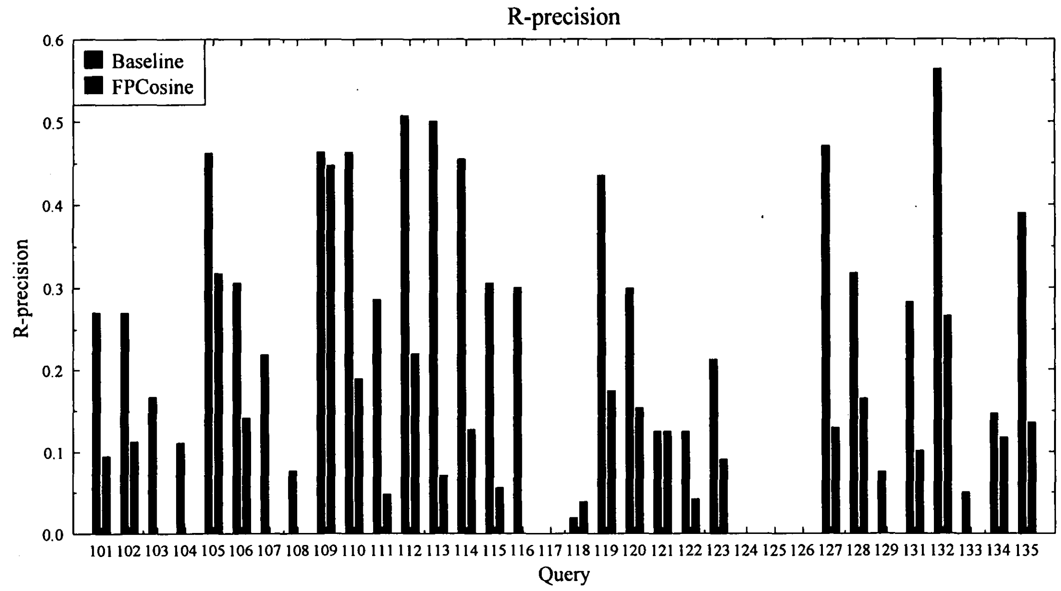


Figure 22: R-precision Values for FP-Cosine Method

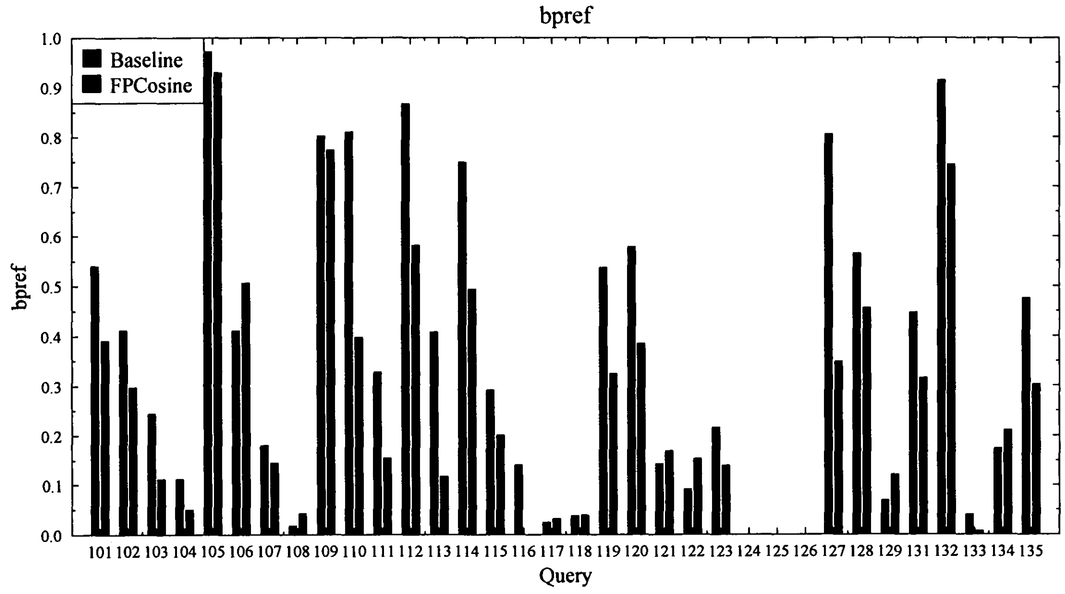


Figure 23: bpref Values for FP-Cosine Method

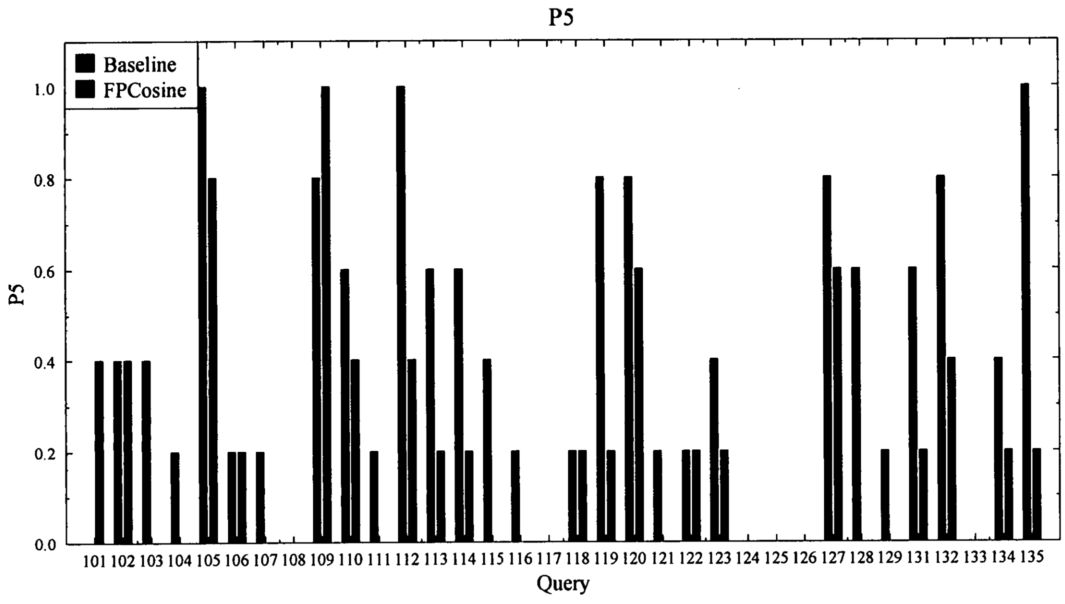


Figure 24: P5 Values for FP-Cosine Method

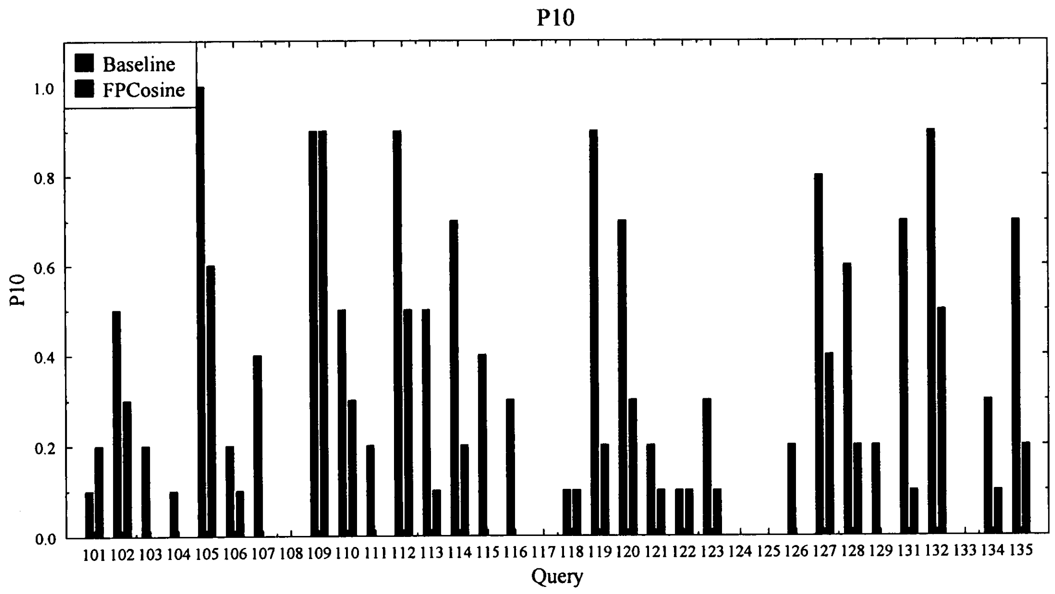


Figure 25: P10 Values for FP-Cosine Method

The complete summary of the results for this method and all observed values for all evaluation measures is shown in Table 5. As shown, in this method, there is no improvement when considering the average values for the all the queries. One reason for this negative result points to the same reason why that Apriori (previous step) algorithm did not show any improvements. These reasons include the lack of correct calculations of the similarities between the assigned concepts in the queries and in the documents.

	Map	R-prec	bpref	P5	P10
Baseline	0.2343	0.2551	0.3646	0.4059	0.4000
FP-Cosine	0.0673 (-71%)	0.0989 (-61%)	0.2626 (-27%)	0.2059 (-49%)	0.1647 (-58%)

Table 5: FP-Cosine Method in compare with Baseline

7.2 Sub-AP:

This section describes the results for our second method, which involves sub-type filtering and weighted query expansion techniques.

I this method, we added a variety of different numbers to the original query. Specifically, we added the top 10 concepts, then the top 20 concepts, and finally the top 40 concepts. Figure 26 shows the results of all three top concept-addition numbers in a single figure.

As we mentioned about the method in chapter 4, this method filtered concepts for each concept subtype. The first subtype was “Disorders”, which can be seen in Figure 26 with the different alpha values. In this case, the alpha value determines the weight between the original baseline technique and the new Sub-AP technique.

In Figure 26, we observe that the Map values behave similarly for all the three top concept-addition numbers. In all cases, the MAP values decrease as the alpha value increases. As shown, the best MAP values are determined at an alpha value of 0.1. The Maximum Map in all the different cases occurred in the Top-40 concept number with an alpha value of 0.1 at a MAP value of 0.2594. The lowest Map value also belongs to the Top-40 concept number with an alpha value of 0.9 at an MAP value of 0.2.

Since all the curves in Figure 26 overlap and behave similarly, we can conclude that using different top concept-addition numbers has no affect on the performance of the information retrieval process. Therefore, adding different numbers of top concepts to the original query has no significant impact.

Figure 27 shows the R-precision values for all the different numbers of top ranked concepts that were added to the original query. This figure has almost the same shape of Figure 26.

The R-Precision value is lowered as the alpha value increases. In this result, the best R-precision value belongs to α=0.1 and the lowest R-precision values occur at α=0.9. In this figure the highest R-precision value occurs at the Top 40 curve with a R-precision value of 0.2972 at α=0.1. The lowest R-precision value is 0.12 at an α=0.9 for Top 20 curve. R-precision values for all the three different numbers of added concepts at similar alpha values are almost the same.

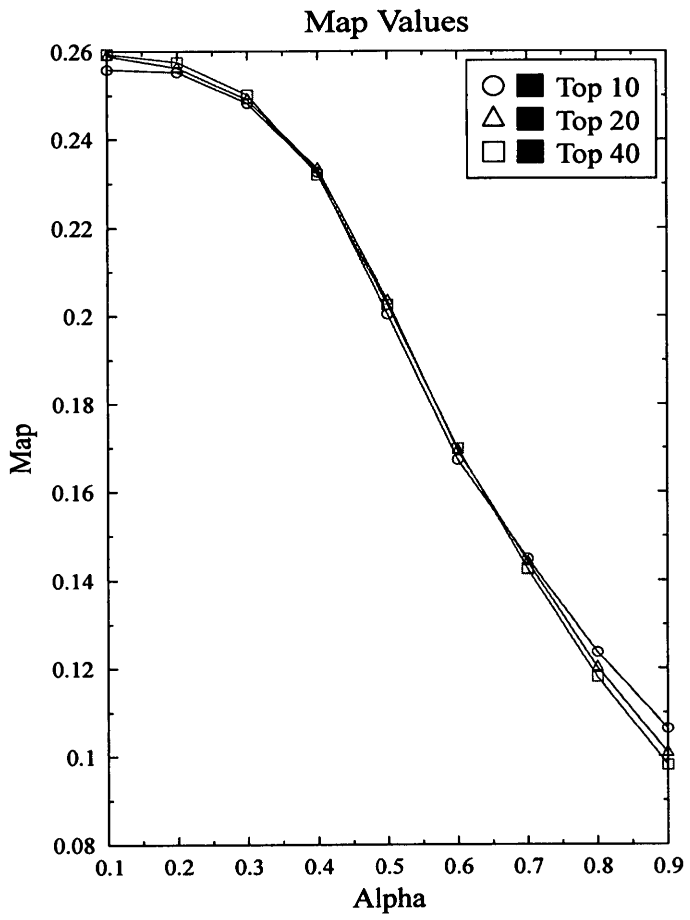


Figure 26: Map Values for Top 10, Top 20 and Top 40 added Concepts for Sub-AP based on α for Disorders

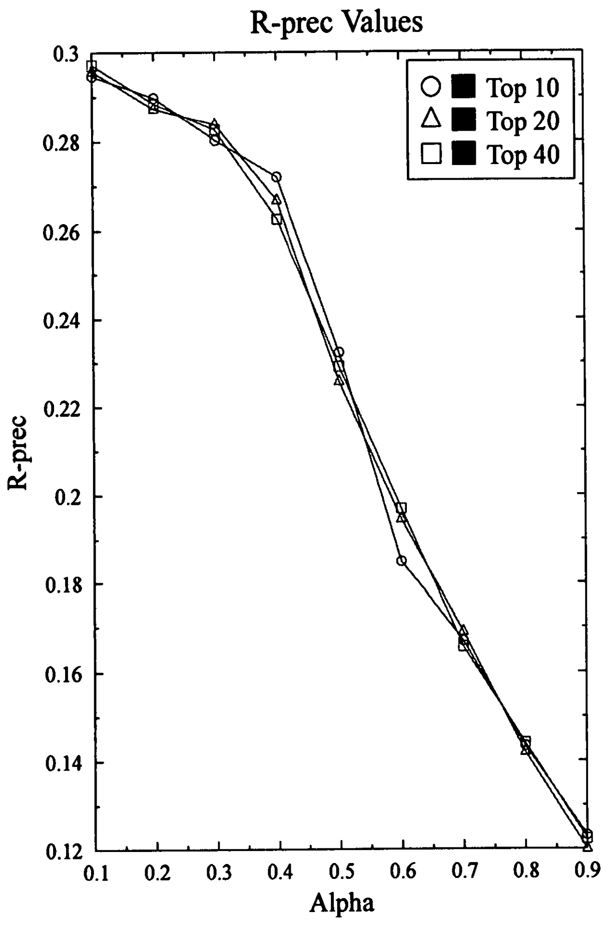


Figure 27: R-precision Values for Top 10, Top 20 and Top 40 added Concepts for Sub-AP based on α for Disorders

Figure 28 shows three Bpref measurements for the three top-number of concepts added. This figure shows that the highest value for Bpref occurs at α=0.1 and Top 20 concepts with the value for Bpref of 0.4269. The shape of this figure is exactly the same as the two previous figures; the Bpref values decrease as the value of alpha increases. The lowest value appears at α=0.1 and Top 10 concepts with a Bpref value of 0.3.

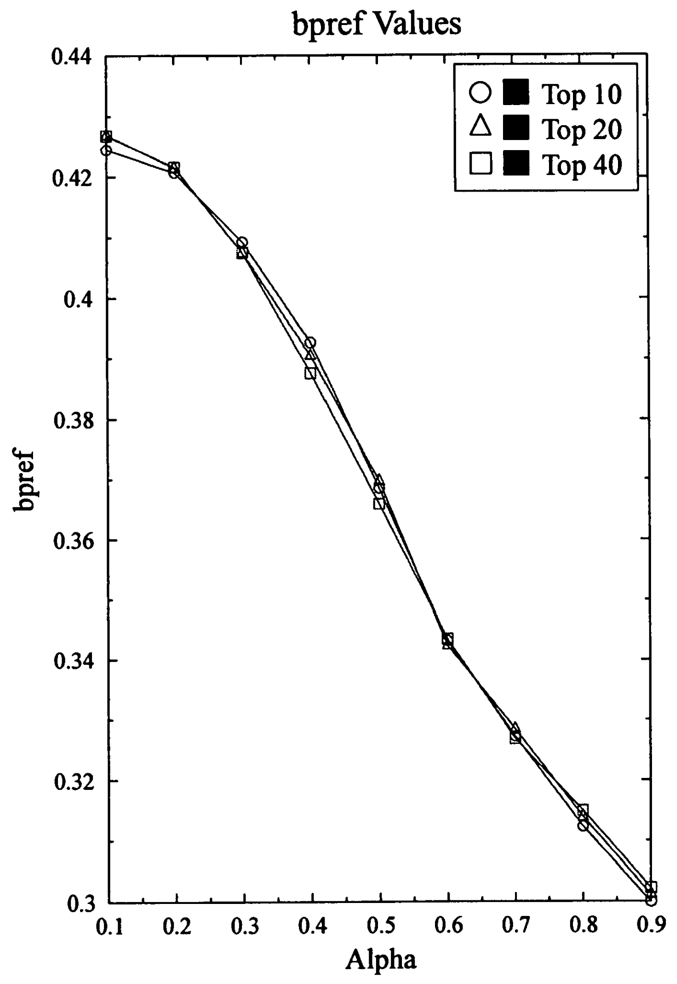


Figure 28: bpref Values for Top 10, Top 20 and Top 40 added Concepts for Sub-AP based on α for Disorders

Figure 29 illustrates the P5 value behaviors for all three different numbers of added concepts to the expanded queries. In various parts of this figure, we observe that same exact P5 values for different values of α. For example, with α = 0.2 and α = 0.3 for Top 10 added concepts, the P5 value remains at 0.4529. The two highest P5 values occur at α = 0.2 for Top 20 and α = 0.3 for Top 40 with the same P5 value of 0.4706. In this figure, unlike the previous figures, we do not observe a clear behavior as the alpha values increases.

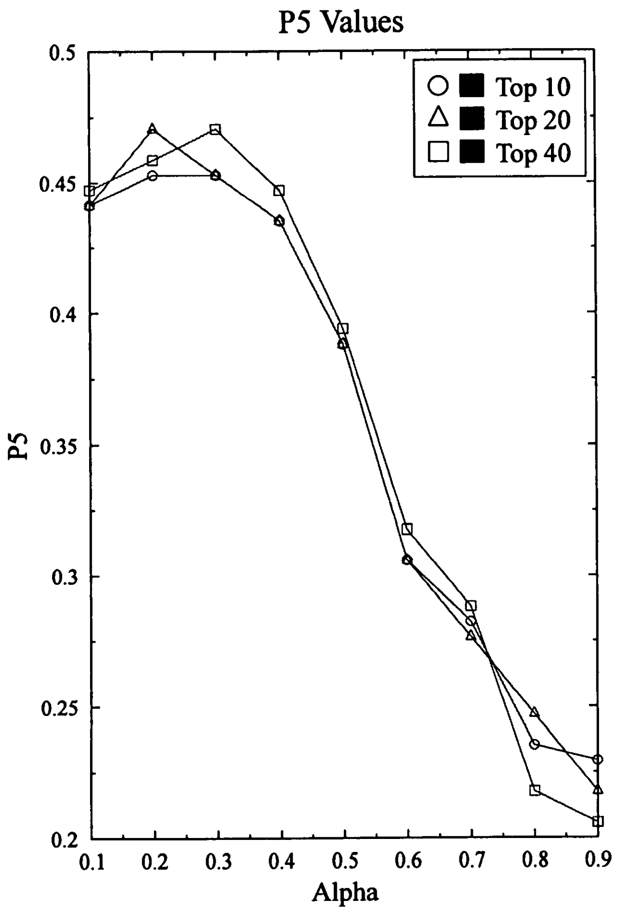


Figure 29: P5 Values for Top 10, Top 20 and Top 40 added Concepts for Sub-AP based on α for Disorders

Figure 30 shows the P10 values for all numbers of top-added concepts. The marvelous feature of this figure is that the highest P10 values occurred at the same exact alpha values (0.3), with slightly different P10 values, with an average value of 0.4176.

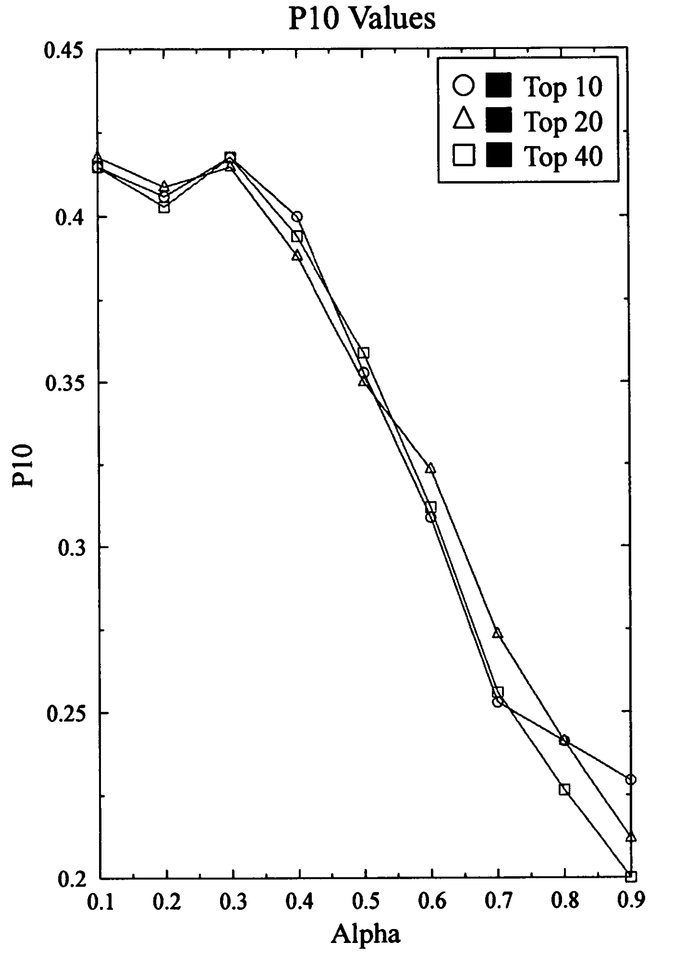


Figure 30: P10 Values for Top 10, Top 20 and Top 40 added Concepts for Sub-AP based on α for Disorders

Table 6 shows the overall highest values for all the different runs (Baseline, Top 10, Top 20, Top 40). The overall highest values occurred at $\alpha = 0.1$ for all the three different number of added concepts. If looking at which top-number of added concepts has the best values, we observe that the Top 40 added concepts shows the best improvements.

	Map	R-prec	Bpref	P5	P10
Baseline	0.2343	0.2551	0.3646	0.4059	0.4000
Top 10	0.2559 (+8%)	0.2947 (+15%)	0.4245 (+16%)	0.4412 (+8%)	0.4147 (+3%)
Top 20	0.259 (+10%)	0.2958 (+15%)	0.4269 (+17%)	0.4412 (+8%)	0.4176 (+4%)
Top 40	0.2594 (+10%)	0.2972 (+16%)	0.4268 (+17%)	0.4471 (+10%)	0.4147 (+3%)

Table 6: Comparison Disorders Subtype Results with baseline

Figure 31 shows the Map values for different numbers of added concepts extracted from the Procedure subtypes. This figure shows that most of the MAP values across different values of α remains the same with different number of added concepts (10, 20, 40). The highest Map value belongs to the Top-10 added number concepts; the difference in this case is very small compared to the other Top values, with an amount of 0.2647.

R-precision values of all the three different number of added concepts are shown in Figure 32. These curves have almost the same pattern until $\alpha = 0.8$. At that point, for $\alpha = 0.9$, the curve for Top 20 peaks at a R-prevision value of 0.5928.

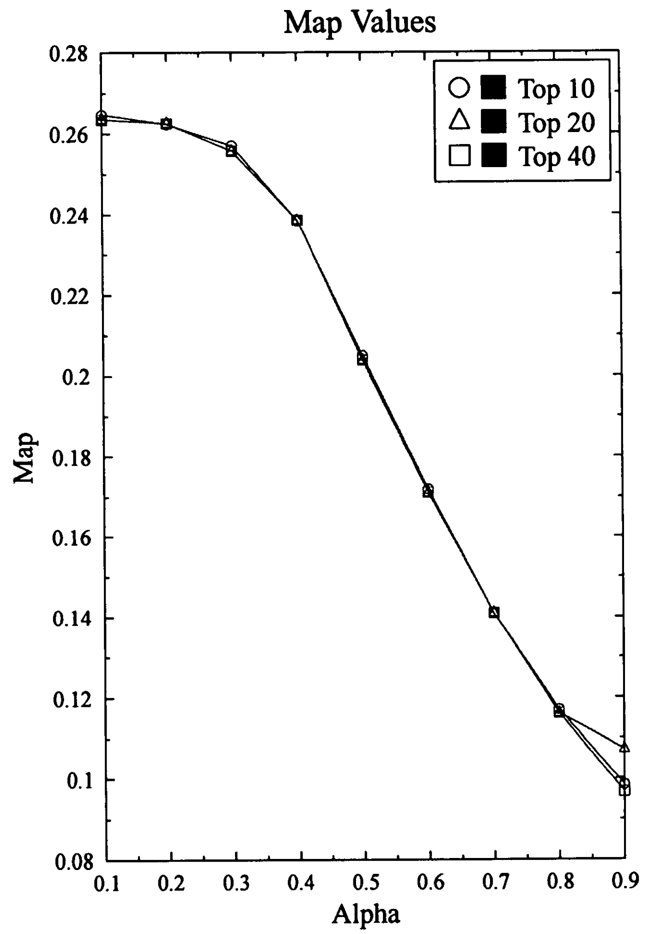


Figure 31: Map Values for Top 10, Top 20 and Top 40 added Concepts for Sub-AP based on α For Procedure

bpref figures (Figure 33) have the same behavior for all the three different number of added concepts. All three figures are going down based on α values. The most valuable bpref for all these three different settings occurred in α = 0.1 for Top 10 with 0.4302.

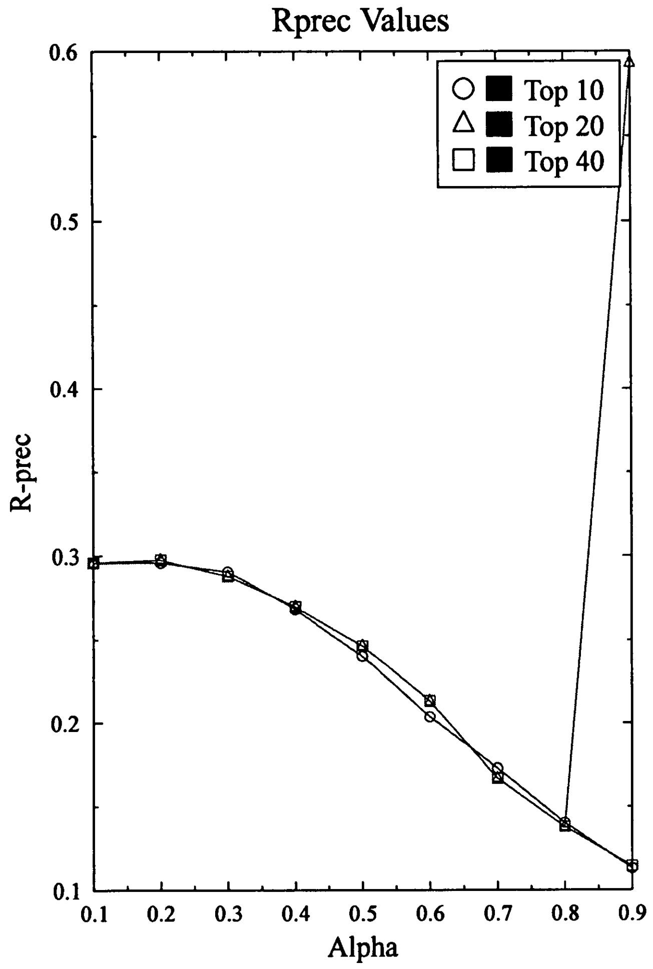


Figure 32: R-precision Values for Top 10, Top 20 and Top 40 added Concepts for Sub-AP based on α For Procedure

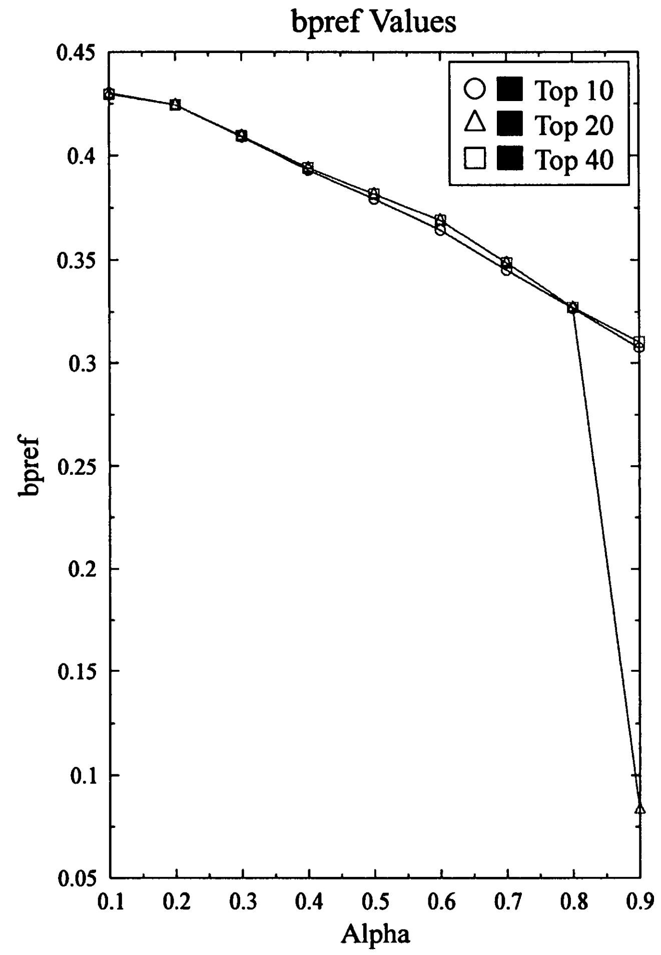


Figure 33: bpref Values for Top 10, Top 20 and Top 40 added Concepts for Sub-AP based on α For Procedure

The P5 values in Figure 34 can be described just like bpref (Figure 33) and Map (Figure 32) figures. The only difference is that the P5 values increases after α = 0.8 in Top 10 and in Top 40. The highest P5 occurs at three times with different α amounts at a P5 value of 0.4706.

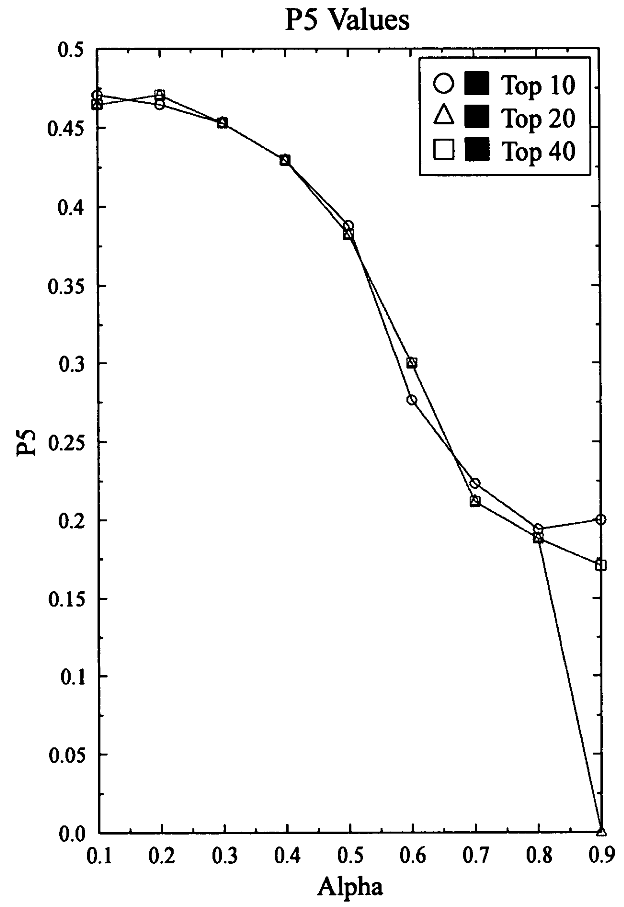


Figure 34: P5 Values for Top 10, Top 20 and Top 40 added Concepts for Sub-AP based on α For Procedure

P10 values in Figure 35 have the same exact pattern as that of P5 in Figure 34. The highest P10 value occurs at α = 0.1 for Top 10 with a P10 value of 0.4206.

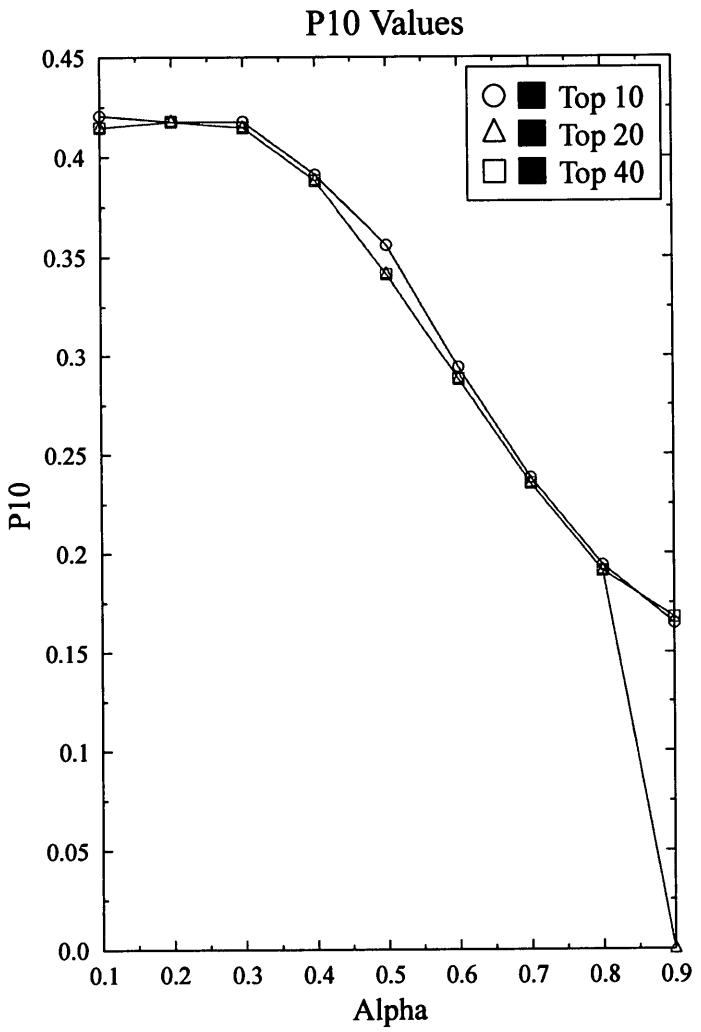


Figure 35: P10 Values for Top 10, Top 20 and Top 40 added Concepts concepts for Sub-AP based on α For Procedure

The overall results for the Procedure subtype is shown in Table 7. The best overall results occur with α = 0.1 and α = 0.2 for all three different numbers of added concepts. Therefore, a comparison can be made between these values and and that of the baseline.

	Map	R-prec	bpref	P5	P10
Baseline	0.2343	0.2551	0.3646	0.4059	0.4000
Top 10 $\alpha = 0.1$	0.2647 (+12%)	0.2959 (+15%)	0.4302 (+17%)	0.4706 (+15%)	0.4206 (+5%)
Top 10 $\alpha = 0.2$	0.2623 (+11%)	0.2961 (+16%)	0.4244 (+16%)	0.4647 (+14%)	0.4176 (+4%)
Top 20 $\alpha = 0.1$	0.2635 (+12%)	0.296 (+16%)	0.4296 (+17%)	0.4647 (+14%)	0.4147 (+3%)
Top 20 $\alpha = 0.2$	0.2625 (+12%)	0.2976 (+16%)	0.4244 (+16%)	0.4706 (+15%)	0.4176 (+4%)
Top 40 $\alpha = 0.1$	0.2635 (+12%)	0.296 (+16%)	0.4296 (+17%)	0.4647 (+14%)	0.4147 (+3%)
Top 40 $\alpha = 0.2$	0.2625 (+12%)	0.2976 (+16%)	0.4244 (+16%)	0.4706 (+15%)	0.4176 (+4%)

Table 7: Comparison Procedure Subtype Results with baseline

7.3 ICD9-Top

This section describes the results for our third method, which is based on using ICD-9 codes and their description in the query expansion process.

7.3.1 ICD9-Top20

Map values comparisons between different queries for the ICD9-Top20 as well as the baseline is shown Figure 36. Almost all the queries have improvement in their MAP values when compared to the baseline. The highest MAP value occurs in query 112 with +26% improvement over the baseline approach. That is a significant improvement.

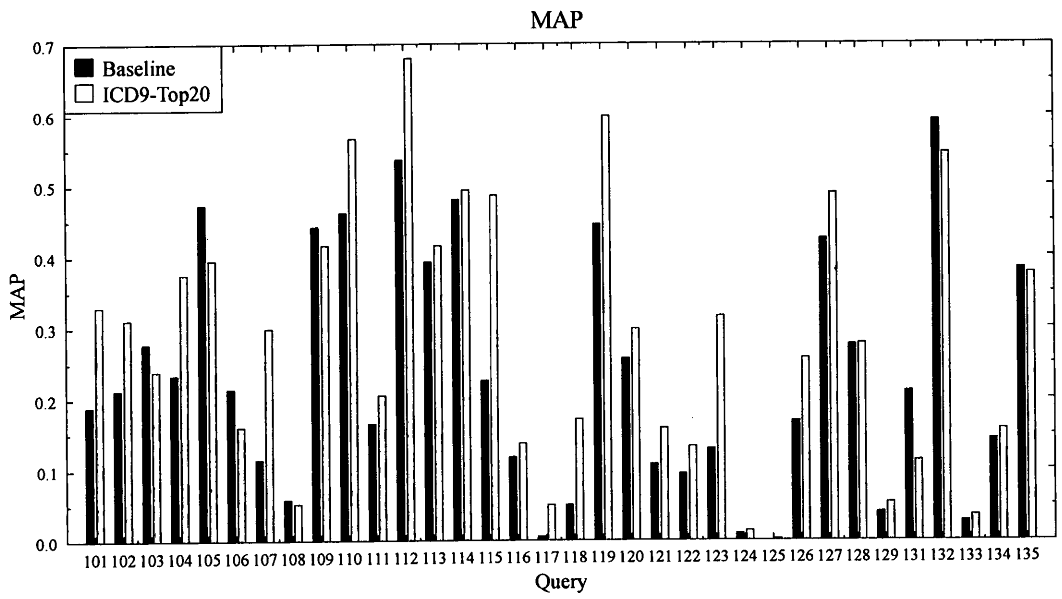


Figure 36: Comparison MAP values for ICD9-TOP20 method and Baseline

Figure 37 shows the R-precision values for both the ICD9-Top 20 method and the baseline method. Most of the queries have an improvement in the R-precision values, which shows the value of this new method.

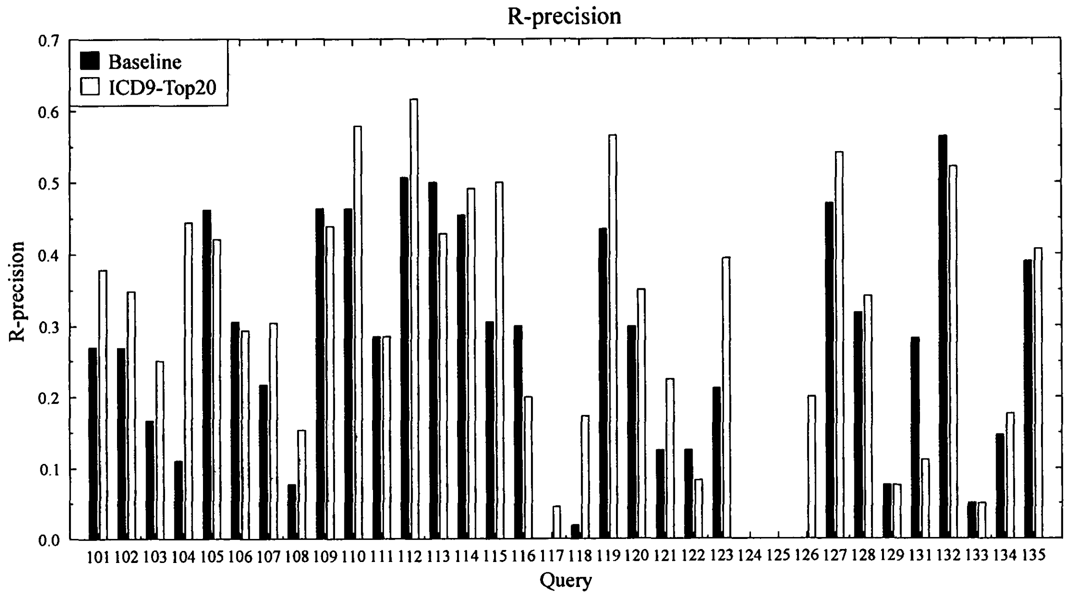


Figure 37: Comparison R-precision values for ICD9-TOP20 method and Baseline

Bpref values for ICD9-Top20 method are compared to that of the baseline in the Figure 38. The Bpref values of the ICD9-Top-20 method are higher, or at worst, equal to that of the baseline.

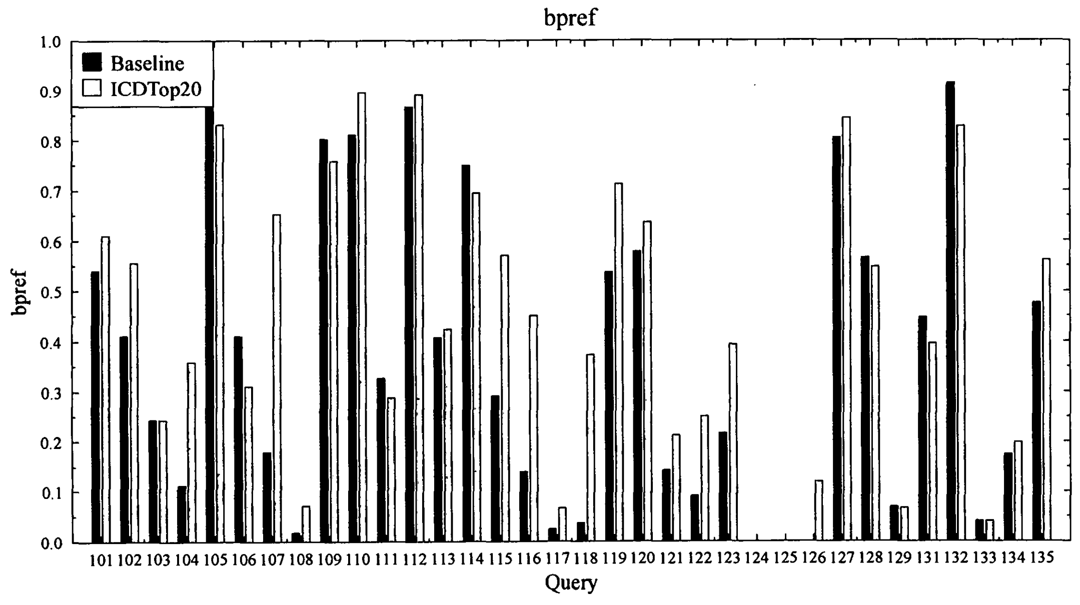


Figure 38: Comparison Bpref values for ICD9-TOP20 method and Baseline

Figure 39 shows that the P5 values for all the queries are at least equal to, or much higher, than the baseline values. The only exceptions are query 105 and query 127. This means that the average of P5 values would indicate a good improvement for all the queries when compared to the baseline.

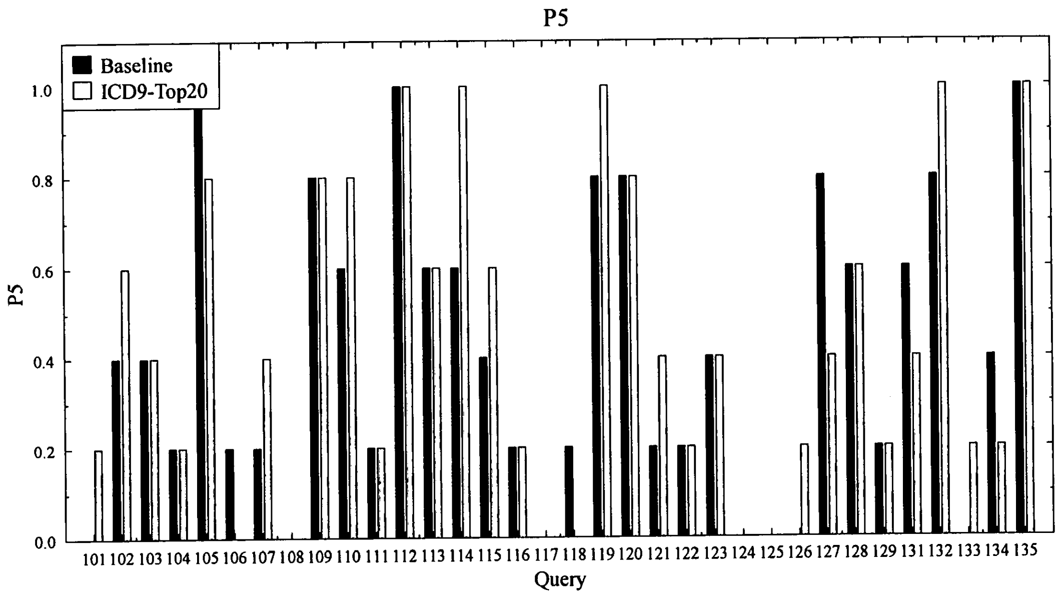


Figure 39: Comparison P5 values for ICD9-TOP20 method and Baseline

Figure 40 shows the P10 values. We observe that it has almost the same behavior as the P5 values in Figure 39. As shown, most of the queries exhibit a P10 value exceeding the baseline values.

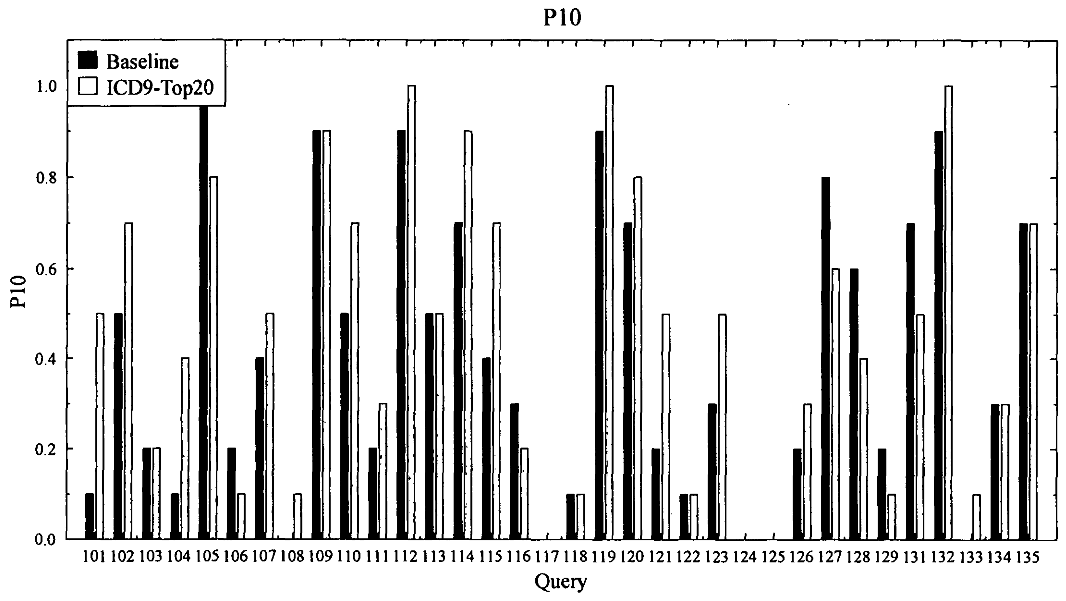


Figure 40: Comparison P10 values for ICD9-TOP20 method and Baseline

Figure 41 and Table 8 show a complete picture of the ICD9-Top20 improvements in all the evaluation measure for all the queries. As shown in Table 8, the MAP and R-precision values with the highest improvement show a 20% improvement compare to the baseline. For the bpref values, P10 values, and P5 values, we observe a 19%, 13% and 7% improvement, respectively. This shows the effectiveness of this method over the standard baseline approach.

	Map	R-precision	Bpref	P5	P10
Baseline	0.2343	0.2551	0.3646	0.4059	0.4000
ICD9-Top20	0.2826 (+20%)	0.3078 (+20%)	0.4366 (+19%)	0.4353 (+7%)	0.4559 (+13%)

Table 8: Comparison ICD9-Top20 Method with Baseline results

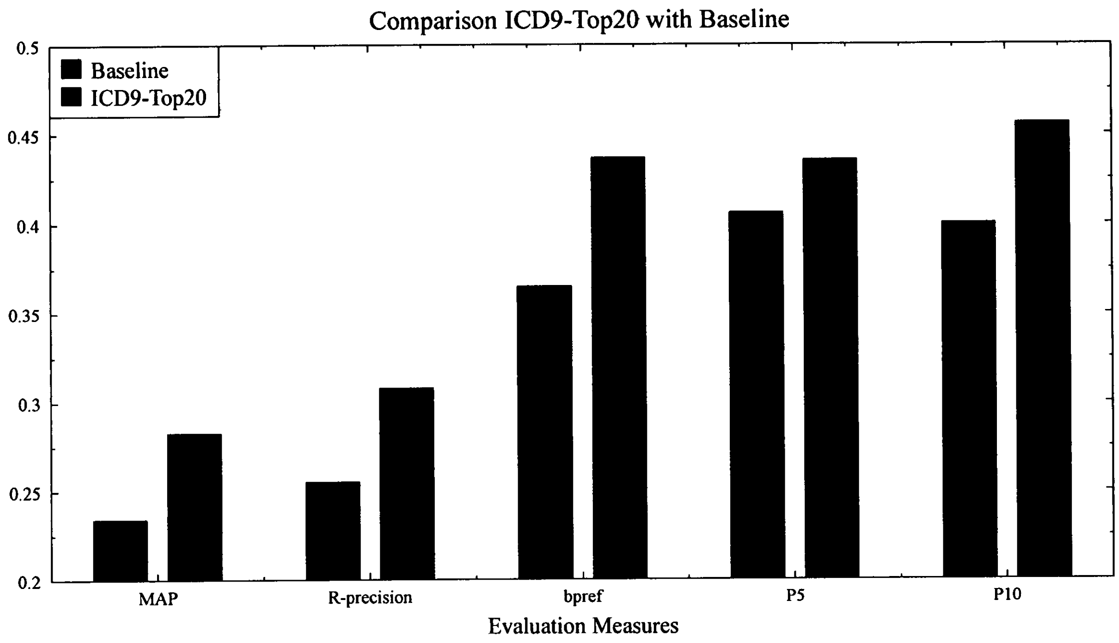


Figure 41: Comparison of Evaluation Measures Between ICD9-Top20 and Baseline for All the Queries

7.3.2 ICD9-Top50

MAP values for the ICD9-Top50 method is shown in Figure 42. In almost all the queries, the MAP values are higher than the baseline values. This shows the positive impact of this powerful method.

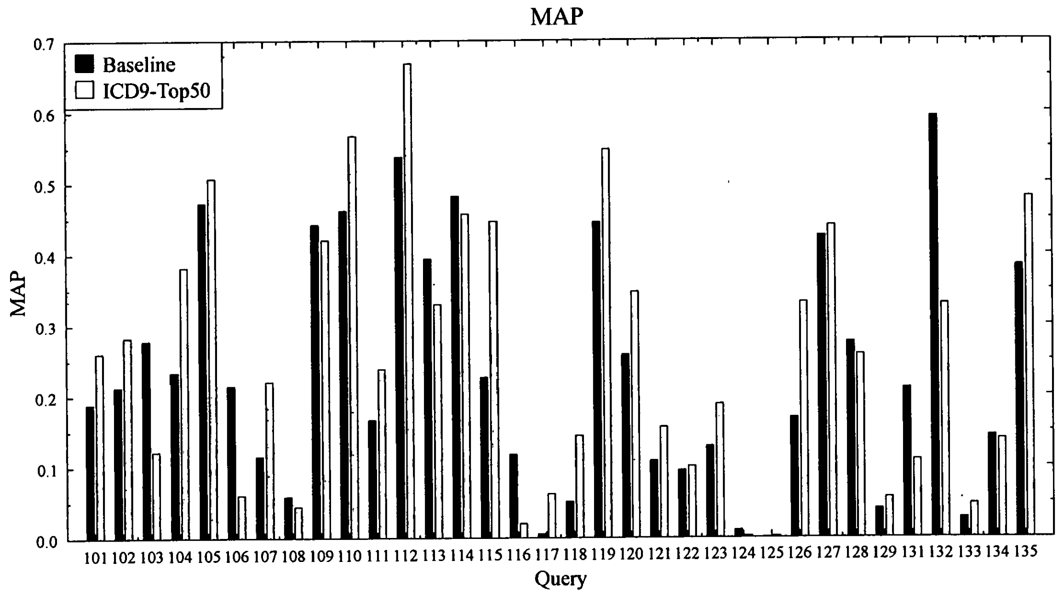


Figure 42: Comparison MAP values for ICD9-TOP50 method and Baseline

Figure 43 shows the R-precision values for each query; the only query in the method that has a lower value than the baseline is query 132. The overall result shows an improvement for all the queries.

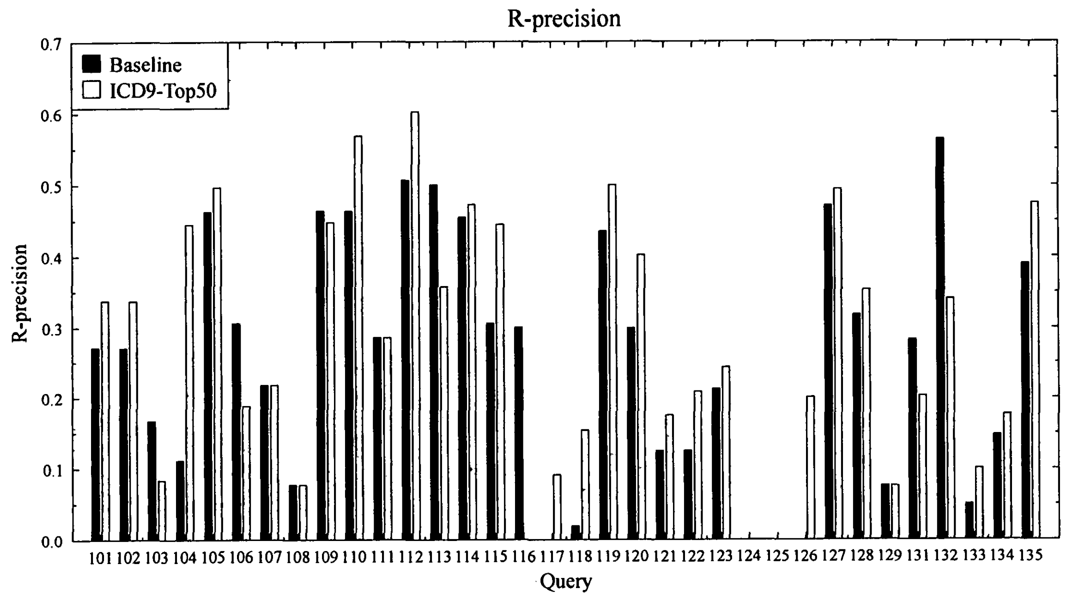


Figure 43: Comparison R-precision values for ICD9-TOP50 method and Baseline

Figure 44 shows to bpref values, and we observe that the ICD9-Top50 method exhibits improvements in almost all of the queries.

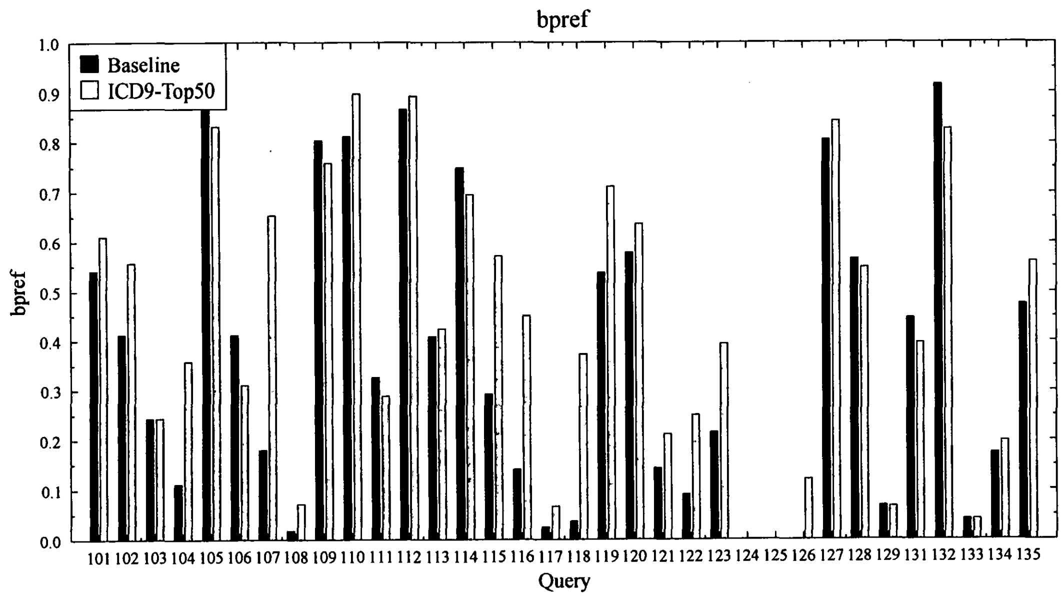


Figure 44: Comparison bpref values for ICD9-TOP50 method and Baseline

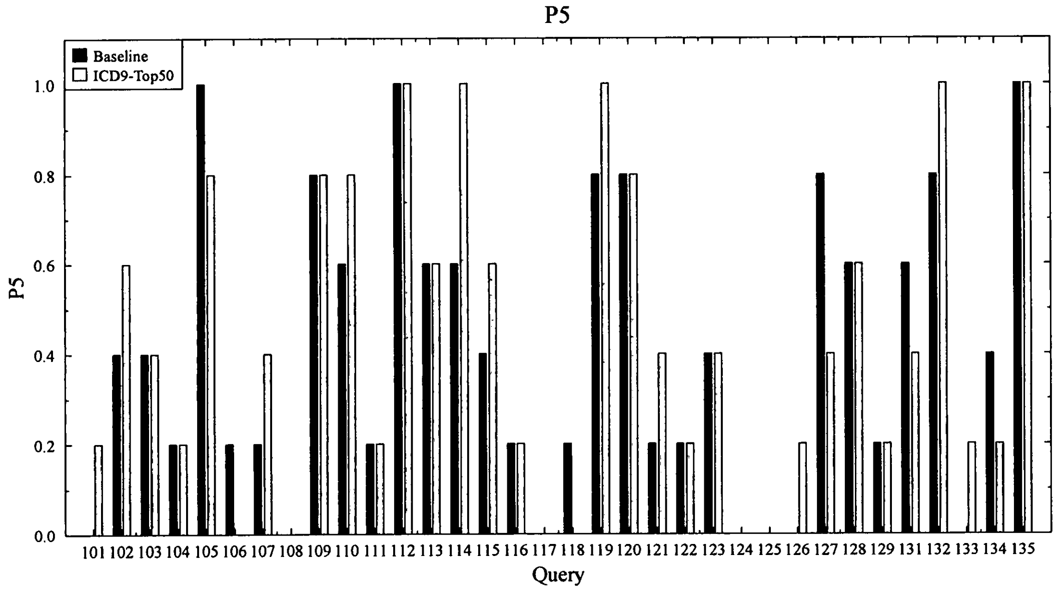


Figure 45: Comparison P5 values for ICD9-TOP50 method and Baseline

Figure 45 shows the P5 values, the differences in P5 values between this method and baseline method. These differences show how the ICD9-Top50 method made improvements in most of the queries. For some queries, the results are equal to the baseline method.

The P10 values are shown in Figure 46. The values in all the queries have improvements in comparison to the values of the baseline. The only exceptions are query 105, query 119 and query 132.

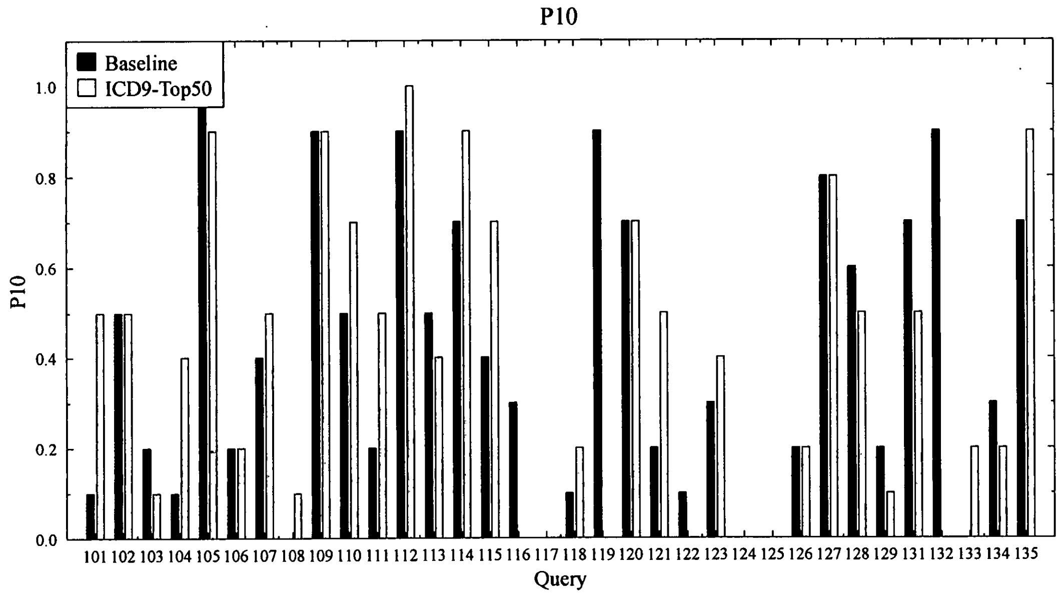


Figure 46: Comparison P10 values for ICD9-TOP50 method and Baseline

The complete results for ICD9-Top50 method are shown in table 9 and Figure 47.

The summary results show that all the evaluation measures point to improvements between 8-13 % compared to the baseline method. The highest improvement in this method is observed in the P5 values, with a 13% improvement when compared to the baseline.

	Map	R-precision	Bpref	P5	P10
Baseline	0.2343	0.2551	0.3646	0.4059	0.4000
ICD9-Top50	0.2577 (+9%)	0.2808 (+10%)	0.3968 (+8%)	0.4588 (+13%)	0.4441 (+11%)

Table 9: Comparison ICD9-Top50 Method with Baseline results

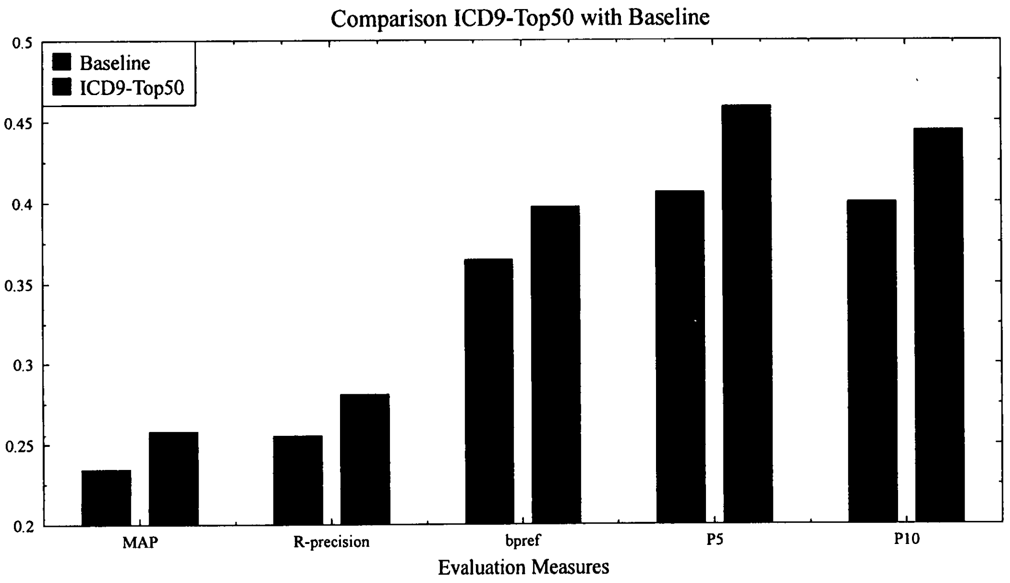


Figure 47: Comparison of Evaluation Measures Between ICD9-Top20 and Baseline for All the Queries

8 Analysis and Discussion

Chapter 7 presented each of the three method's results in detail. In this chapter, we will analyze and dissect the results of the methods; we will compare the three methods and discuss the best choice for improving medical information retrieval.

8.1 APFP-Cosine Discussion

According to the evaluation measure from chapter 7, it is evident that the APFP-Cosine method did not show any improvements in any of the evaluation measures (section 7.1). The APFP-Cosine method uses a combination of the Apriori algorithm, the FP-Growth algorithm, and the Cosine Similarity measurement. One possible reason for the ineffectiveness of the APFP-Cosine algorithm is the use of the Cosine Similarity measurements between the expanded queries and each report. Using each expanded query's concepts to find the most similar reports might have given us the reports that contained those same concepts from the expanded queries. However, it is possible that many of the relevant reports do not contain those exact concepts as mentioned in the expanded queries. For this reason, this method was not effective. Relying on exact matches does not work well with the TREC Medical Data track.

The poor performance of the APFP-Cosine is shown in its evaluation scores. Specifically, the AP-Cosine and the FP-Cosine method performed -59% and -61%, respectively, worse than the baseline in the R-Precision measurements. By all evaluation measures, it is shown than the APFP-Cosine method is ineffective.

8.2 Sub-AP Discussion

The second method, Sub-AP (Section 7.2), had improvements in all the experimental settings we used. This shows that the consideration of specific subtypes of the concepts can make the information retrieval results more accurate. For comparison purposes, we will only consider the best results from the AP-Sub method. Therefore, we look at the case where the Top 10 added concepts to the original query with the $\alpha = 0.1$ for AP-Sub method for Procedures. Specifically, the AP-Sub improved R-precision and bpref over the baseline by 15% and 17%, respectively.

8.3 ICD9-Top Discussion

The last method, which has improvements in both the ICD9-Top20 and ICD9-Top50 ranked reports, is the ICD9-Top. Between the usage of the Top 20 and the Top 50 ranked reports, the Top 20 ranked reports had better results. We will consider only the ICD9-Top20 in order to compare with other results. Specifically, the ICD9-Top20 improved by 13% over the baseline in the P10 measurements. It also improved by 19% and by 20% in the bpref and the R-Precision measures, respectively.

8.4 Improved Methods Compared to the Baseline

Figure 48 compares the best results of the Sub-AP method, the ICD9-Top20 method, as well as the baseline method. This comparison illustrates that the most effective method is ICD9-Top20, which has the highest evaluation measure in in all measurements, except for P5. In the P5 measurement, the Sub-AP method for Procedures outperforms the ICD9-Top20 method by 8%, and the baseline method by 15%.

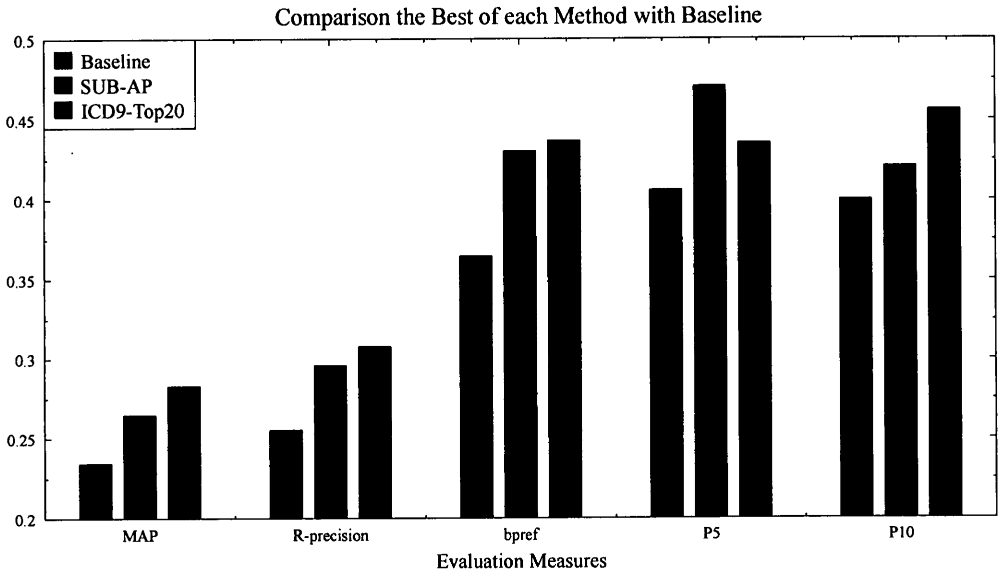


Figure 48: Comparison of all Evaluation Measures Between Sub-AP, ICD9-Top20 and Baseline

8.5 The Significance Test

After analyzing each of our proposed methods and comparing the results with the baseline, we need to know how much more improved our methods are. In order to reach this goal and avoid the influence of uncontrollable noises which can happen because of better performance of some topics compared to others, we applied a significant test for the best results in each of our two-best methods.

To choose the correct test we need to have knowledge about the test that suits our dataset. Our samples are correlated and we need a nonparametric test. The methods are based on repeated-measures. For each measurement, we use two samples: one is the baseline result and the other is our new methods' results. For our purposes, we chose the Wilcoxon-signed rank tests. The null hypothesis asserts that the medians of the two samples are identical. In the following tables, the significant tests have been done on the best results of our proposed methods. The first test is shown in tables 10, 11, 12, 13 and 14, which belong to the ICD9-Top20 approach. Here, we see that all measurements show significance, except for P5. Table 15, 16, 17, 18 and 19 show the test results for ICD9-Top50 evaluation measures. Here we see that the Map and R-prec measures show significant improvements.

The significant test of the best run of the Sub-AP approach which is for Procedure in $\alpha = \mathbf{0.1}$ and for Top 10 added concepts are shown in Table 20, 21, 22, 23 and 24. The results of the test shows that the bpref measure in this run have significant improvements compared to the baseline.

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-3.0517*. The p-value is *0.00114*. The result **is significant** at $p \leq 0.01$.	The Z-value is *-3.0517*. The p-value is *0.00228*. The result **is significant** at $p \leq 0.01$.
	The W-value is *119*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *119*. The distribution is approximately normal. Therefore, the Z-value above should be used.
Significance Level = 0.05	The Z-value is *-3.0517*. The p-value is *0.00114*. The result **is significant** at $p \leq 0.05$	The Z-value is *-3.0517*. The p-value is *0.00228*. The result **is significant** at $p \leq 0.05$
	The W-value is *119*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *119*. The distribution is approximately normal. Therefore, the Z-value above should be used.

Table 10: The results of significant test for ICD9-Top20 (Map Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-2.8651*. The p-value is *0.00205*. The result **is significant** at $p \leq 0.01$.	The Z-value is *-2.8651*. The p-value is *0.0041*. The result **is significant** at $p \leq 0.01$.
	The W-value is *85*. The critical value of W for *N=29* at $p \leq 0.01$ *is 110*. Therefore, the result **is significant** at $p \leq 0.01$.	The W-value is *85*. The critical value of W for *N=29* at $p \leq 0.01$ *is 100*. Therefore, the result **is significant** at $p \leq 0.01$.
Significance Level = 0.05	The Z-value is *-2.8651*. The p-value is *0.00205*. The result **is significant** at $p \leq 0.05$.	The Z-value is *-2.8651*. The p-value is *0.0041*. The result **is significant** at $p \leq 0.05$.
	The W-value is *85*. The critical value of W for *N=29* at $p \leq 0.05$ *is 140*. Therefore, the result **is significant** at $p \leq 0.05$.	The W-value is *85*. The critical value of W for *N=29* at $p \leq 0.05$ *is 126*. Therefore, the result **is significant** at $p \leq 0.05$.

Table 11: The results of significant test for ICD9-Top20 (R-prec Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-2.7253*. The p-value is *0.00317*. The result **is significant** at $p \leq 0.01$.	The Z-value is *-2.7253*. The p-value is *0.00634*. The result **is significant** at $p \leq 0.01$.
	The W-value is *100*. The critical value of W for *N=30* at $p \leq 0.01$ *is 120*. Therefore, the result **is significant** at $p \leq 0.01$.	The W-value is *100*. The critical value of W for *N=30* at $p \leq 0.01$ *is 109*. Therefore, the result **is significant** at $p \leq 0.01$.
Significance Level = 0.05	The Z-value is *-2.7253*. The p-value is *0.00317*. The result **is significant** at $p \leq 0.05$.	The Z-value is *-2.7253*. The p-value is *0.00634*. The result **is significant** at $p \leq 0.05$.
	The W-value is *100*. The critical value of W for *N=30* at $p \leq 0.05$ *is 151*. Therefore, the result **is significant** at $p \leq 0.05$.	The W-value is *100*. The critical value of W for *N=30* at $p \leq 0.05$ *is 137*. Therefore, the result **is significant** at $p \leq 0.05$.

Table 12: The results of significant test for ICD9-Top20 (bpref Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-1.0178*. The p-value is *0.15386*. The result is not significant at p ≤ 0.01.	The Z-value is *-1.0178*. The p-value is *0.30772*. The result is not significant at *p ≤ 0.01*.
	The W-value is *55*. The critical value of W for *N=17* at *p ≤ 0.01 is 27*. Therefore, the result is not significant at *p ≤ 0.01*.	The W-value is *55*. The critical value of W for *N=17* at *p ≤ 0.01 is 23*. Therefore, the result is not significant at *p ≤ 0.01*.
Significance Level = 0.05	The Z-value is *-1.0178*. The p-value is *0.15386*. The result is not significant at *p ≤ 0.05*.	The Z-value is *-1.0178*. The p-value is *0.30772*. The result is not significant at *p ≤ 0.05*.
	The W-value is *55*. The critical value of W for *N=17* at *p ≤ 0.05 is 41*. Therefore, the result is not significant at *p ≤ 0.05*.	The W-value is *55*. The critical value of W for *N=17* at *p ≤ 0.05 is 34*. Therefore, the result is not significant at *p ≤ 0.05*.

Table 13: The results of significant test for ICD9-Top20 (P5 Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-1.7857*. The p-value is *0.3673*. The result is not significant at p ≤ 0.01.	The Z-value is *-1.7857*. The p-value is *0.07346*. The result is not significant at *p ≤ 0.01*.
	The W-value is *87.5*. The critical value of W for *N=24* at *p ≤ 0.01 is 69*. Therefore, the result is not significant at *p ≤ 0.01*.	The W-value is *87.5*. The critical value of W for *N=24* at *p ≤ 0.01 is 61*. Therefore, the result is not significant at *p ≤ 0.01*.
Significance Level = 0.05	The Z-value is *-1.7857*. The p-value is *0.03673*. The result **is significant** at *p ≤ 0.05*.	The Z-value is *-1.7857*. The p-value is *0.07346*. The result is not significant at *p ≤ 0.05*.
	The W-value is *87.5*. The critical value of W for *N=24* at *p ≤ 0.05 is 91*. Therefore, the result **is significant** at *p ≤ 0.05*.	The W-value is *87.5*. The critical value of W for *N=24* at *p ≤ 0.05 is 81*. Therefore, the result is not significant at *p ≤ 0.05*.

Table 14: The results of significant test for ICD9-Top20 (P10 Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-1.7011*. The p-value is *0.04457*. The result is not significant at $p \leq 0.01$.	The Z-value is *-1.7011*. The p-value is *0.08914*. The result is not significant at $p \leq 0.01$.
	The W-value is *198*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *198*. The distribution is approximately normal. Therefore, the Z-value above should be used.
Significance Level = 0.05	The Z-value is *-1.7011*. The p-value is *0.04457*. The result **is significant** at $p \leq 0.05$	The Z-value is *-1.7011*. The p-value is *0.08914*. The result is not significant at $p \leq 0.05$.
	The W-value is *198*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *198*. The distribution is approximately normal. Therefore, the Z-value above should be used.

Table 15: The results of significant test for ICD9-Top50 (MAP Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-1.7306*. The p-value is *0.04182*. The result is not significant at $p \leq 0.01$.	The Z-value is *-1.7306*. The p-value is *0.08364*. The result is not significant at $p \leq 0.01$.
	The W-value is *127*. The critical value of W for *N=28* at $p \leq 0.01$ *is 101*. Therefore, the result is not significant at $p \leq 0.01$.	The W-value is *127*. The critical value of W for *N=28* at $p \leq 0.01$ *is 91*. Therefore, the result is not significant at $p \leq 0.01$.
Significance Level = 0.05	The Z-value is *-1.7306*. The p-value is *0.04182*. The result **is significant** at $p \leq 0.05$	The Z-value is *-1.7306*. The p-value is *0.08364*. The result is not significant at $p \leq 0.05$.
	The W-value is *127*. The critical value of W for *N=28* at $p \leq 0.05$ *is 130*. Therefore, the result **is significant** at $p \leq 0.05$.	The W-value is *127*. The critical value of W for *N=28* at $p \leq 0.05$ *is 116*. Therefore, the result is not significant at $p \leq 0.05$.

Table 16: The results of significant test for ICD9-Top50 (R-prec Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-1.3089*. The p-value is *0.0951*. The result is not significant at $p \leq 0.01$.	The Z-value is *-1.3089*. The p-value is *0.1902*. The result is not significant at $p \leq 0.01$.
	The W-value is *194*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *194*. The distribution is approximately normal. Therefore, the Z-value above should be used.
Significance Level = 0.05	The Z-value is *-1.3089*. The p-value is *0.0951*. The result is not significant at $p \leq 0.05$.	The Z-value is *-1.3089*. The p-value is *0.1902*. The result is not significant at $p \leq 0.05$.
	The W-value is *194*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *194*. The distribution is approximately normal. Therefore, the Z-value above should be used.

Table 17: The results of significant test for ICD9-Top50 (bpref Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-1.5896*. The p-value is *0.05592*. The result is not significant at $p \leq 0.01$.	The Z-value is *-1.5896*. The p-value is *0.11184*. The result is not significant at $p \leq 0.01$.
	The W-value is *49*. The critical value of W for *N=18* at $p \leq 0.01$ *is 32*. Therefore, the result is not significant at $p \leq 0.01$.	The W-value is *49*. The critical value of W for *N=18* at $p \leq 0.01$ *is 27*. Therefore, the result is not significant at $p \leq 0.01$.
Significance Level = 0.05	The Z-value is *-1.5896*. The p-value is *0.05592*. The result is not significant at $p \leq 0.05$.	The Z-value is *-1.5896*. The p-value is *0.11184*. The result is not significant at $p \leq 0.05$.
	The W-value is *49*. The critical value of W for *N=18* at $p \leq 0.05$ *is 47*. Therefore, the result is not significant at $p \leq 0.05$.	The W-value is *49*. The critical value of W for *N=18* at $p \leq 0.05$ *is 40*. Therefore, the result is not significant at $p \leq 0.05$.

Table 18: The results of significant test for ICD9-Top50 (P5 Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-1.0494*. The p-value is *0.14686*. The result is not significant at $p \leq 0.01$.	The Z-value is *-1.0494*. The p-value is *0.29372*. The result is not significant at $p \leq 0.01$.
	The W-value is *123.5*. The critical value of W for *N=25* at $p \leq 0.01$ *is 76*. Therefore, the result is not significant at $p \leq 0.01$.	The W-value is *123.5*. The critical value of W for *N=25* at $p \leq 0.01$ *is 68*. Therefore, the result is not significant at $p \leq 0.01$.
Significance Level = 0.05	The Z-value is *-1.0494*. The p-value is *0.14686*. The result is not significant at $p \leq 0.05$.	The Z-value is *-1.0494*. The p-value is *0.29372*. The result is not significant at $p \leq 0.05$.
	The W-value is *123.5*. The critical value of W for *N=25* at $p \leq 0.05$ *is 100*. Therefore, the result is not significant at $p \leq 0.05$.	The W-value is *123.5*. The critical value of W for *N=25* at $p \leq 0.05$ *is 89*. Therefore, the result is not significant at $p \leq 0.05$.

Table 19: The results of significant test for ICD9-Top50 (P10 Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-1.3848*. The p-value is *0.08379*. The result is not significant at $p \leq 0.01$.	The Z-value is *-1.3848*. The p-value is *0.16758*. The result is not significant at $p \leq 0.01$.
	The W-value is *203*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *203*. The distribution is approximately normal. Therefore, the Z-value above should be used.
Significance Level = 0.05	The Z-value is *-1.3848*. The p-value is *0.08379*. The result is not significant at $p \leq 0.05$.	The Z-value is *-1.3848*. The p-value is *0.16758*. The result is not significant at $p \leq 0.05$.
	The W-value is *203*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *203*. The distribution is approximately normal. Therefore, the Z-value above should be used.

Table 20: The results of significant test for Sub-AP method based on the Top 10 added concepts in $\alpha = 0.1$ for Procedure subtype (MAP Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-1.3914*. The p-value is *0.08226*. The result is not significant at $p \leq 0.01$.	The Z-value is *-1.3914*. The p-value is *0.16452*. The result is not significant at $p \leq 0.01$.
	The W-value is *177*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *177*. The distribution is approximately normal. Therefore, the Z-value above should be used.
Significance Level = 0.05	The Z-value is *-1.3914*. The p-value is *0.08226*. The result is not significant at $p \leq 0.05$.	The Z-value is *-1.3914*. The p-value is *0.16452*. The result is not significant at $p \leq 0.05$.
	The W-value is *177*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *177*. The distribution is approximately normal. Therefore, the Z-value above should be used.

Table 21: The results of significant test for Sub-AP method based on the Top 10 added concepts in $\alpha = 0.1$ for Procedure subtype in (R-prec Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-2.5618*. The p-value is *0.00523*. The result **is significant** at $p \leq 0.01$.	The Z-value is *-2.5618*. The p-value is *0.01046*. The result is not significant at $p \leq 0.01$.
	The W-value is *127*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *127*. The distribution is approximately normal. Therefore, the Z-value above should be used.
Significance Level = 0.05	The Z-value is *-2.5618*. The p-value is *0.00523*. The result **is significant** at $p \leq 0.05$.	The Z-value is *-2.5618*. The p-value is *0.01046*. The result **is significant** at $p \leq 0.05$.
	The W-value is *127*. The distribution is approximately normal. Therefore, the Z-value above should be used.	The W-value is *127*. The distribution is approximately normal. Therefore, the Z-value above should be used.

Table 22: The results of significant test for Sub-AP method based on the Top 10 added concepts in $\alpha = 0.1$ for Procedure subtype in (bpref Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-1.1688*. The p-value is *0.121*. The result is not significant at $p \leq 0.01$.	The Z-value is *-1.1688*. The p-value is *0.242*. The result is not significant at $p \leq 0.01$.
	The W-value is *90.5*. The critical value of W for *N=22* at $p \leq 0.01$ *is* *55*. Therefore, the result is not significant at $p \leq 0.01$.	The W-value is *90.5*. The critical value of W for *N=22* at $p \leq 0.01$ *is* *48*. Therefore, the result is not significant at $p \leq 0.01$.
Significance Level = 0.05	The Z-value is *-1.1688*. The p-value is *0.121*. The result is not significant at $p \leq 0.05$.	The Z-value is *-1.1688*. The p-value is *0.242*. The result is not significant at $p \leq 0.05$.
	The W-value is *90.5*. The critical value of W for *N=22* at $p \leq 0.05$ *is* *75*. Therefore, the result is not significant at $p \leq 0.05$.	The W-value is *90.5*. The critical value of W for *N=22* at $p \leq 0.05$ *is* *65*. Therefore, the result is not significant at $p \leq 0.05$.

Table 23: The results of significant test for Sub-AP method based on the Top 10 added concepts in $\alpha = 0.1$ for Procedure subtype in (P5 Values)

	One-tailed	Two-tailed
Significance Level = 0.01	The Z-value is *-0.6458*. The p-value is *0.25785*. The result is not significant at $p \leq 0.01$.	The Z-value is *-0.6458*. The p-value is *0.5157*. The result is not significant at $p \leq 0.01$.
	The W-value is *138.5*. The critical value of W for *N=25* at $p \leq 0.01$ *is* *76*. Therefore, the result is not significant at $p \leq 0.01$.	The W-value is *138.5*. The critical value of W for *N=25* at $p \leq 0.01$ *is* *68*. Therefore, the result is not significant at $p \leq 0.01$.
Significance Level = 0.05	The Z-value is *-0.6458*. The p-value is *0.25785*. The result is not significant at $p \leq 0.05$.	The Z-value is *-0.6458*. The p-value is *0.5157*. The result is not significant at $p \leq 0.05$.
	The W-value is *138.5*. The critical value of W for *N=25* at $p \leq 0.05$ *is* *100*. Therefore, the result is not significant at $p \leq 0.05$.	The W-value is *138.5*. The critical value of W for *N=25* at $p \leq 0.05$ *is* *89*. Therefore, the result is not significant at $p \leq 0.05$.

Table 24: The results of significant test for Sub-AP method based on the Top 10 added concepts in $\alpha = 0.1$ for Procedure subtype in (P10 Values)

9 Conclusion and Future Work

9.1 APFP-Cosine Method

In this method, we tried to combine query expansion with Cosine Similarity measures in order to re-rank the best possible results. Unfortunately, this method did not help the information retrieval task in any of the evaluation measures.

Using Cosine Similarity is based on mapping each of the concepts in the query with the reports. The poor performance of this method shows that having a strong similarity between the query terms and the report content is not necessarily a strong evidence to judge the relevancy.

9.2 AP-Sub Method

In this method, we used the Apriori algorithm in order to perform query expansion. Next, we used subtype filtering in order to improve the retrieval results. This method showed improvements across all measurements in comparison to the baseline.

Applying Sub-type filtering can help us separate the relevant concepts from the irrelevant ones based on the given query. In other words, this method helps make the important concepts more noticeable and more valuable in the result. For this reason, we were able to see better retrieval results in this method.

9.3 ICD9-Top

During our research work, we noticed that ICD-9 codes that appeared in the reports gave an important insight to the content in the entire report. These codes are unique and could be determined easily. These important characteristics of the TREC medical reports led us to new query expansion techniques based on ICD9 codes from the top retrieved documents in the baseline.

This method, which can be defined as using query expansion techniques based on the most frequent ICD-9 codes, showed improvements in all of the measures in comparison to the baseline. It was also the method with the best results among the three proposed methods in this research paper.

9.4 Future Work

An overview of all the methods and results show that the most effective way to retrieve the most accurate results belong to the ones that combine the top ranked baseline reports and query expansion techniques. This research work shows that using the baseline retrieval results is a good place to start. I believe that the work presented here is in its infancy, and has a long way to go. As more and more medical work is done electronically, more data becomes available to researchers, and that will lead to better medical information retrieval techniques.

The next step for this research should focus on other aspects of ICD-9 codes, using a variety of different IR methods. We also like to work on less-focused parts of the medical reports which other researches have ignored previously. Combining different sub-type filtering techniques in order to create new methods can be a new direction for this work in future research. Finally, we would also like to work with physicians in the industry to further improve the results and create a user interface in the near future.

Bibliography

[1]. A. Savasere, E. O. (1995). An Efficient Algorithm for Mining Assocation Rules in Large Databases. *Proc Int'l Conf Very Large Data Bases*, (pp. 432-444,). Zurich.

[2]. A. R. Aronson, F. Lang. (2010). An overview of MetaMap: historical perspective and recent advances. *Am Med Inform Assoc* , 229-236.

[3]. A. Tangpong, A. R. (n.d.). Applying Association Rules Discovery in Query Expansion Process.

[4]. B. Andreopoulos, X. H. (2009). Promoting Diversity in Top Hits for Biomedical Passage Retrieval. Springer-Verlag .

[5]. B. He, J. X. (2010). Modeling term proximity for probabilistic information retrieval models. *Information Sciences* .

[6]. B. King, L. Wang, I. Provalov, J. Zhou. (2011). Cengage Learning at TREC 2011 Medical Track. *TREC.*

[7]. C. Buckley, G. S. (2005). Automatic Query Expansion Using SMART: TREC 3. *TREC* . TREC.

[8]. C. D. Manning, P. R. (2008). Introduction to Information Retrieval. Cambridge University Press.

[9]. C. L. A. Clarke, F. S. (2005). The TREC 2005 Terabyte Track. TREC.

[10]. Cha, S. (n.d.). Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions . *INTERNATIONAL JOURNAL OF MATHEMATICAL MODELS AND METHODS IN APPLIED SCIENCES* .

[11]. Chris Buckley, Ellen M. Voorhees. (2004). Retrieval Evaluation with Incomplete Information. *SIGIR 04.* Sheffield: ACM.

[12]. D. Zhu, B. Carterette. (2012). Combining Multi-level Evidence for Medical Record Retrieval. *ACM* .

[13]. Ellen M. Voorhees, William Hersh. (2012). Overview of the TREC 2012 Medical Records Track. *TREC.*

[14]. G. Salton, C. B. (1998). Improving Retrieval Performance by Relevance Feedback. *Journal of American Society for Information Science* .

[15]. Iadh Ounis, G. A. (2005). Terrier Information Retrieval Platform. *Springer* .

[16]. J. Xu, W. B. (1996). Query Expansion Using Local and Global Document Analysis. *SIGIR96.* ACM.

[17]. J. Lewis, S. Ossowski, J. Hicks, M. Errami, H. R. Garner. (2006). Text similarity: an alternative way to search MEDLINE. *Oxford University Press* .

[18]. J. Leveling, L. Goeuriot, L. Kelly, G. J. F. Jones. (2012). TREC .

[19]. J. S. Park, M. S. (1995). Efficient Parallel Data Mining for Association Rules. *Conf Information and Knowledge Management.*

[20]. Jinxi Xu, W. Bruce Croft. (2009). *Improving the Effectiveness of Information Retrieval with Local Context Analysis.* Massachusetts: ACM.

[21]. K. Foh. *Integrating Healthcare: The Role and Value of Mobile Operators in eHealth.*

[22]. K. J. Hannah, M. J. Ball. *Health Informatics.* USA: Springer.

[23]. L. Ballesteros, B. C. (1997). Phrasal Translation and Query Expansion Techniques for Cross-Language Information Retrieval. *SIGIR97.* ACM.

[24]. Liu, Y. (2010). Study on Application of Apriori Algorithm in Data Mining. *IEEE.*

[25]. M. Daoud, D. K. (2011). York University at TREC 2011: Medical Records Track . *TREC.* TREC .

[26]. M. Lupu, J. H. (2009). TREC-CHEM: Large Scale Chemical Information Retrieval Evaluation at TREC. *SIGIR.* ACM.

[27]. M. Mitra, A. S. (1998). Improving Automatic Query Expansion . *SIGIR 98.* ACM.

[28]. Measures Appendixes. Trec 16.

[29]. N. Limsopatham, C. M. (2004). Learning to Selectively Rank Patients' Medical History. *CIKM'13.*

[30]. N. Limsopatham, C. Macdonald, I. Ounis, G. McDonald, M. Bouamrane. (2011). University of Glasgow at Medical Records Track 2011: Experiments with Terrier. TREC.

[31]. PWC. *The emerging benefits of electronic medical record use in community-based care.* PWC. PWC.

[32]. Q. T. Zeng, D. R. (2012). Synonym, Topic Model and Predicate-Based Query Expansion for Retrieving Clinical Documents. *AMIA.*

[33]. R. Agrawal, J. S. (1995). *Parallel Mining of Association Rules: Design, Implementation, and Experience.* IBM Research Report RJlOG4.

[34]. R. Agrawal, R. S. (1994). Fast Algorithms for Mining Association Rules. nternational Conference on VLDB .

[35]. R. Attar, A.S. Fraenkel. (1977). Local feedback in full-text retrieval systems. *ACM* , 397–417.

[36]. R. Haux . (2006). Health information systems — past, present, future. *Medical Informatics* .

[37]. S. Bedrick, T. Edinger, A. Cohen, W. Hersh. (2012). Identifying Patients for Clinical Studies from Electronic Health Records: TREC 2012 Medical Records Track at OHSU. TREC.

[38]. S. Bhattacharya, C. G. Harris, Y. Mejova, Ch. Yang, P. Srinivasan. (2011). The University of Iowa at TREC 2011: Microblogs, Medical Records and Crowdsourcing. TREC.

[39]. Singhal, A. (2001). Modern Information Retrieval: A Brief Overview.

[40]. Voorhees, E. M. Query Expansion using Lexical-Semantic Relations .

[41]. W. B. Croft, D. J. Harper. (1979). Using Probabilistic Models of Document Retrieval Without Relevance Information. *Journal of Documentation* , 285-295.

[42]. X. Xu, W. Zhu, X. Zhang, X. Hu, I. Song. (2006). A Comparison of Local Analysis, Global Analysis and Ontology-based Query Expansion Strategies for Bio-medical Literature Search. *IEEE.*

[43]. X. Yin, J. X. (2011). A Survival Modeling Approach to Biomedical Search Result Diversification Using Wikipedia.

[44]. X. Yin, J. X. (2010). Mining and Modeling Linkage Information from Citation Context for Improving Biomedical Literature Retrieval.

[45]. Y. Zhao, G. Karypis. (2002). Improve Precategorized Collection Retrieval by Using Supervised Term Weighting Schemes. *IEEE.*

[46]. Y. Qiu, H. F. (1993). Concept Based Query Expansion . *SIGIR93.* ACM.

[47]. Z. Ye, J. X. (2011). Finding a Good Query-Related Topic for Boosting Pseudo Relevance Feedback. *Journal of the American Society for Information Science and Technology (JASIST), , 62*, 748-760.

[48]. Z. Ye, J. X. (2012). Mining Multilingual Association Dictionary from Wikipedia for Cross-Language Information Retrieval. *Journal of the American Society for Information Science and Technology (JASIST).*

A. Scripts for Methodoligies

In each chapter we described different programs in order to achieve specific results. These programs have been written in different languages which are Java, Phyton and Perl. In this section the most important ones are listed in the following.

A.1 Generating Report's Vector

In Chapter 3 we described the Report's Vector which created in order to calculate the cosine similarity. The following program has been used in order to generate the Report Vectors.

```
import java.io*
import java.util.ArrayList;
import java.util.HashMap;

//0 20051127OP-cQsnkGImzZbN-848-71049104 c2599456 3 term0 Nt=29623 TF=88525 @{0
31714398 0} T201 PHYS,
//TF = 3
//IDF  = LOG (95708/Nt)
//              Nt = 29623

public class ProcessingVector101 {

      public static void main(String[] args) throws IOException {

            //XXX - GLOBAL VARIABLE - XXX
            double totalNumberOfReports = 95708;

            FileWriter vector101;
            BufferedWriter out;

            //OUTPUT/Users/Hoda/Documents/workspace/VisitID
            vector101 = new
FileWriter("/Users/Hoda/Documents/workspace/VisitID/vectors101T.txt");
            out = new BufferedWriter(vector101);
            out.write("<CONCEPT> <TF-IDF>\n");

            //INPUT 1 --->
```

```
			FileInputStream fstream = new
FileInputStream("/Users/Hoda/Downloads/BM25b0.75_0.res");
			// Get the object of DataInputStream
			DataInputStream in = new DataInputStream(fstream);
			BufferedReader br = new BufferedReader(new InputStreamReader(in));
			String strLine;

			//Array was made here
			HashMap<String, Integer> hm = new HashMap();

			while ((strLine = br.readLine())  !=  null     )
			{
				if(strLine.startsWith("101"))
					continue;

				if(strLine.startsWith("102"))
				{
//System.out.println("Reading line " + strLine);
					String[] f1Array = strLine.split(" ");
					String reportID = f1Array[2];

		System.out.println("Reading reportID " + reportID);

					//reportIDs[cnt++] = reportID;
					hm.put(reportID, 0);

				}
				else
				{
System.out.println("Not found 101. Moving On!");
					break;
				}

			}


			//INPUT 2 --->
			FileInputStream fstream2 = new
FileInputStream("/Users/Hoda/Downloads/trecmed_docname_docid_conid_tf_termid_Ndo
c_ttf_types.txt");
			// Get the object of DataInputStream
			DataInputStream in2 = new DataInputStream(fstream2);
			BufferedReader br2 = new BufferedReader(new
InputStreamReader(in2));
			String strLine2;

			int i=0;
			while((strLine2 = br2.readLine()) != null)
			{
				//System.out.println("Line: " + i++);
				String[] f2Array = strLine2.split(" ");
				String comparedToReportID = f2Array[1];

				boolean answer = hm.containsKey(comparedToReportID);

				if(answer==true)
				{

					//System.out.println("Report ID Found in " +
strLine2);

					String concept = f2Array[2];

					double TF = Integer.parseInt(f2Array[3]);


					String ntString = f2Array[5];
					ntString = ntString.replaceFirst("Nt=", "");
```

```
                                double ntInt = Integer.parseInt(ntString);
                                double IDF = Math.log10(totalNumberOfReports/ntInt);

                                double TFIDF = TF*IDF;

                                out.write(concept + " " + TFIDF);
                                out.flush();
                                out.write("\n");
                                System.out.println("Wrote to file: " + concept + " "
+ TFIDF + "==================");


                        }
                }
        }
}
```

A.2 Generating Query's Vector

In Chapter 3 the Query's Vector generated in order to calculate Cosine Similarity. The following program has been used to create each Query's vector.

```
import java.awt.Dimension;
import java.io.*
import java.util.Arrays;


public class VectorFor101 {

        public static void main(String[] args) throws IOException {

                //GLOBAL VARIABLE
                float totalNumberOfReports = 95708;

                FileWriter vector101;
                BufferedWriter out;

                //OUTPUT
                vector101 = new FileWriter("OUT");
                out= new BufferedWriter(vector101);
                out.write("<CONCEPT> <SUPPORT>\n");

                //INPUT 1 ---> 101 C003075
                FileInputStream fstream = new FileInputStream("INPUT1");
                // Get the object of DataInputStream
                DataInputStream in = new DataInputStream(fstream);
                BufferedReader br = new BufferedReader(new InputStreamReader(in));
                String strLine;
//INPUT 2 ---> 0 20051127OP-cQsnkGImzZbN-848-71049104 c2599456 3 term0
Nt=29623 TF=88525 @{0 31714398 0} T201 PHYS,
             FileInputStream fstream2 = new FileInputStream("INPUT2");
             // Get the object of DataInputStream
             DataInputStream in2 = new DataInputStream(fstream2);
             BufferedReader br2 = new BufferedReader(new
InputStreamReader(in2));
             String strLine2;


             while ((strLine = br.readLine())  !=  null     )
             {
                    if(strLine.startsWith("101"))
                    {
                           System.out.println("Reading line " + strLine);
                           String[] f1Array = strLine.split(" ");
                           String concept = f1Array[1];
                           System.out.println("Reading concept " + concept);

                           while((strLine2 = br2.readLine()) != null)
                           {
                                  if(strLine2.contains(concept))
                                  {
             System.out.println(concept + " was found in " + strLine2);
                                         String[] f2Array = strLine2.split(" ");
                                         String ntString = f2Array[5];
                                         ntString = ntString.replaceFirst("Nt=",
"");
                                         int ntInt = Integer.parseInt(ntString);
                           System.out.println("Nt found and parsed with value "
+ ntInt);
                                         float support = (ntInt) /
(totalNumberOfReports);
                                         out.write(concept + " " + support);
System.out.println("Wrote to file: " + concept + " " +   support +
"==================");
                                         break;
                                  }
                           }
                    }
                    else
                    {
                           System.out.println("Not found 101. Now Exiting!");
                           System.exit(0);
                    }
             }
             out.close();
             br.close();
             br2.close();
      }
}
```

A.3 Cosine Similarity

Calculating Cosine Similarity between query vector and report vector (Chapter 3) has been done in this program:

```
package vector;

import java.io.*
import java.text.DecimalFormat;

public class CosineCreation {

        public static void main(String[] args) throws IOException {


                double sumSquareSupport = 0.05000095468826281;



                //OUTPUT
                FileWriter vector101;
                BufferedWriter out;
                vector101 = new FileWriter("/Users/Hoda/Documents/Cosine-
Index/cosine135I.txt");
                out= new BufferedWriter(vector101);
                //out.write("<ReportID> <Cosine>\n");

//**INPUT 1 ---> sort /Users/Hoda/Documents/Vectors-Index/vectors101II.txt >
/Users/Hoda/Documents/Vectors-Index/vectors101IIS.txt
FileInputStream fstream = new FileInputStream("/Users/Hoda/Documents/Vectors-
Index/vectors135IIS.txt");
                // Get the object of DataInputStream
                DataInputStream in = new DataInputStream(fstream);
                BufferedReader br = new BufferedReader(new InputStreamReader(in));
                String strLine;

                String processingID = "";

                double sum=0.0;
                double TFIDF=0.0;
                double sumSquareTFIDF=0.0;

                System.out.println("Start!!");
                while ((strLine = br.readLine())  !=  null     )
                {
                        System.out.println("101TT: Reading line " + strLine);
                        String[] f1Array = strLine.split(" ");
                        String currentID = f1Array[0];
                        String concept = f1Array[1];
                        String TFIDFString = f1Array[2];

                        //System.out.println(currentID + " " + concept + " " +
TFIDFString);

                        TFIDF = Double.parseDouble(TFIDFString);
                        sumSquareTFIDF+=Math.pow(TFIDF, 2);

                        //System.out.println(TFIDF);

                        if(processingID.equals(currentID)==false)
```

```
                    {
                            if(processingID.equals("")==false)
                            {
                                    //Territory of New ID now
                                    //String sumToString = Double.toString(sum);
                                    double bottom =
Math.sqrt((sumSquareTFIDF+sumSquareSupport));
                                    double finalValue = sum/bottom;
                                    //String finalToString =
Double.toString(finalValue);
                                    //System.out.println(new
DecimalFormat("#.########################").format(finalValue));

                                    out.write(new
DecimalFormat("#.########################").format(finalValue));
                                    //System.out.println("sum.toString = " +
sumToString);

                                    out.write("\n");
                                    sumSquareTFIDF=0.0;
                                    sum=0.0;
                                    out.write(currentID);
                                    out.write(" ");
                                    out.flush();
                                    processingID = currentID;
                                    //out.flush();
                                    //System.exit(0);
                            }
                            else
                            {
                                    //Territory of First time, will enter here
only one time
                                    processingID = currentID;
                                    out.write(currentID + " ");
                            }

                    }


                    double support=0.0;

                    //INPUT 2 --->
                    FileInputStream fstream2 = new
FileInputStream("/Users/Hoda/Documents/Vectors-Index/vectors135I.txt");
                    // Get the object of DataInputStream
                    DataInputStream in2 = new DataInputStream(fstream2);
                    BufferedReader br2 = new BufferedReader(new
InputStreamReader(in2));
                    String strLine2;

                    //SKIP FIRST LINE IN STRLINE
                    strLine2 = br2.readLine();

                    while ((strLine2 = br2.readLine())  !=  null   )
                    {
                            //System.out.println("101: Reading line " +
strLine2);


        if(strLine2.toLowerCase().contains(concept.toLowerCase()))
                            {
                                    System.out.println("Found!!!");
                                    String[] f2Array = strLine2.split(" ");
                                    String supportString = f2Array[1];

                                    support = Double.parseDouble(supportString);
//System.out.println("\tSupport = " + supportString + " " + support);
                                    //System.out.println("TFIDF = " + TFIDF);
                                    sum+=(support*TFIDF);
```

```
                                        //System.out.println("sum = " + sum);
                                }
                        }
                }

                double bottom = Math.sqrt((sumSquareTFIDF+sumSquareSupport));
                String finalToString = Double.toString(sum/bottom);
                out.write(finalToString);
                //System.out.println("sum.toString = " + sumToString);
                out.write("\n");
                sumSquareTFIDF=0.0;
                out.flush();
                System.out.println("Phew!!");
//SORT THE OUTPUT sort -nrk2 /Users/Hoda/Documents/Vectors-
Apriori/cosine101I.txt > /Users/Hoda/Documents/Vectors-Index/cosine101IS.txt
        }
```

A.4 Subtype Filtering

In order to find specific subtypes concepts in the whole indexed file (Chapter 4) we used the program which can be seen in this section.

```
import java.io.*;
import java.util.*;


public class Subtype {

        /**
         * @param args
         */
        public static void main(String[] args) {

                FileWriter kstream;
                BufferedWriter out = null;
                ///Users/Hoda/Desktop/serialized.txt
                try{
                        // Create file
                        kstream = new FileWriter("out.txt");
                        out = new BufferedWriter(kstream);
                        //out.write("Hello Java");
                        //Close the output stream
                        //out.close();
                }catch (Exception e){//Catch exception if any
                        System.err.println("Error: " + e.getMessage());
                }


                try{
                        List<String[]> rowList = new ArrayList<String[]>();

                        Hashtable<String, Integer> numbers = new Hashtable<String,
Integer>();


                        int lineNumber = 0, lineNumber2=0;
                        // Open the file that is the first
                        // command line parameter
```

```
                    FileInputStream fstream = new
FileInputStream("/Users/Hoda/Desktop/visit_concept.txt");

                    // Get the object of DataInputStream
                    DataInputStream in = new DataInputStream(fstream);
                    BufferedReader br = new BufferedReader(new
InputStreamReader(in));
                    String strLine;
                    //Read File Line By Line

                    rowList.clear();
                    while ((strLine = br.readLine())!=null)
                    {
                           rowList.clear();
                           lineNumber++;
                           lineNumber2=0;
                           String splittedLine[] = strLine.split(",");
                           rowList.add(splittedLine);
                           String mainVisitID = splittedLine[0];

                           if(numbers.containsKey(mainVisitID))
                                  continue;

                           numbers.put(mainVisitID, 0);

                           //Date before = new Date();

                           //System.out.println("Line Number 1 = " +
lineNumber);

FileInputStream fstream2 = new
FileInputStream("/Users/Hoda/Desktop/visit_concept2.txt");
                           DataInputStream in2 = new DataInputStream(fstream2);
BufferedReader br2 = new BufferedReader(new InputStreamReader(in2));
                           String strLine2;

                           while ( (strLine2 = br2.readLine())!=
null )
                           {
                                  lineNumber2++;

                                  //if(lineNumber2%1000==0)
                                  //System.out.println("Line Number 2 = " +
lineNumber2);

                                  if(lineNumber2==lineNumber)
                                         continue;


                                  String splittedLine2[] = strLine2.split(",");
                                  String guestVisitID = splittedLine2[0];


                                  if(mainVisitID.equals(guestVisitID))
                                  {
                                         //System.out.println("Found it!");
                                         //if(mainVisitID.equals("5K+e6BhC2E6v"))
                                         //{
                                         //System.out.println("THIS IS IT");
//System.out.println("WE are about to REMOVE -------------> " +
splittedLine2[0]);
                                         //}
                                         splittedLine2[0] = "";
                                         rowList.add(splittedLine2);
//System.out.println("mainVisitID = " + mainVisitID + "\tguestVisitID = " +
guestVisitID);
                                  }
```

```
                                    }
//write the rowList to file. rowList also includes all the other copies of
visitID's concepts.
                                    int rowSize = rowList.size();

                                    for(int a=0;a<rowSize;a++)
                                    {
                                          String rowElement[] = rowList.get(a);
                                          int detailArrayLength = rowElement.length;

                                          for(int b=0;b<detailArrayLength;b++)
                                          {
                                                out.write(rowElement[b] + ",");
                                          }

                                    }
                                    out.write("\n");

                                    //if(lineNumber==2 && lineNumber2==5)
                                    //    System.exit(0);

                                    //Date after = new Date();
                                    //long diff = after.getTime() - before.getTime();
                                    //System.out.println("diff = " + diff);
                                    //if(lineNumber==5)
                                    //{
                                    //    System.exit(0);
                                    //}
                                    //rowList.clear();
                              }

                              System.out.println("Done! line number = " + lineNumber);

                              if(strLine==null)
                                    System.out.println("It is Null");

                              //Close the input stream
                              in.close();
                              out.close();
                        }catch (Exception e){//Catch exception if any
                              System.err.println("Error: " + e.getMessage());
                        }


            }

      }
```

A.5 Perl Program for Re-reanking

The re-ranking step in the section 4 has been done by Perl in the following program:

```
# ///////////////////////////////////////////////////

#!/usr/bin/perl

$alpha=$ARGV[0];
print "alpha=$alpha\n";

#---------------------------------------------
$indexQueryFile="/media/backup/Rule propagation/101-level1-2-queryContext-
0.79.txt";

$indexDocFile="/media/backup/visitCindex_docname_docno_c_tf_types.txt";
$baseline="/media/backup/BM25b0.75_baseline_visits_terms_concepts.res";


# query list
$listQueries="101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116
117 118 119 120 121 122 123 124 125 126 127 128 129 131 132 133 134 135";

my @listq = split(' ', $listQueries);
my @listd = split(' ', $lisnbdoc);

$out="/media/backup/Rule propagation/Vector-Reranking_0.79_level1-2-
alpha$alpha.res";
open(OUT, ">$out") or die("Unable to open file $out");

open(QFILE, $indexQueryFile) or die("Unable to open file");
# read file into an array
@indexQuery = <QFILE>;
close(QFILE);

%docconcepts=();
%tf=();
%ndoc=();
open(DFILE, $indexDocFile) or die("Unable to open file");
foreach $line (<DFILE>)
{
     @dataline=split(/ /, $line);
       $visid=$dataline[0];
       $concept=$dataline[2];
       $docconcepts{$visid}=$docconcepts{$visid}.$concept." ";
       $tf{$concept.$visid}=$dataline[3];
       $ndoc{$concept.$visid}=$dataline[5];
}
close(DFILE);

open(BFILE, $baseline) or die("Unable to open file");
# read file into an array
@indexBaseline = <BFILE>;
close(BFILE);

$Ntotaldoc=17267;

foreach my $q (@listq)
{
    print "processing query $q ****************\n";
    # extract the concepts of the query
    #---------------------------------------------------------------------
----
    %QConW=();
    $NormQ=0;
    foreach $lineQ (@indexQuery)
    {
         #print "query line $lineQ";
         chomp ($line);
         @dataline=split(/ /, $lineQ);
         $qcand=$dataline[0];
         if($qcand eq $q)
              {
                   $QConW{$dataline[1]}=$dataline[2];
                  $NormQ+=$dataline[2]**2;
                  #print "$q $dataline[1] $dataline[2]\n";
              }
    }

    #print "NormQ $NormQ\n";

    # read the collection document index file and extract the list of documents,
then for each document, extract the concepts of the document in a hash table
    #---------------------------------------------------------------------------
----
    %newscore=();
    %Normdoc=();
    %baselinescore=();
    $maxbaselinescore=0;
    foreach $linebase (@indexBaseline)
    {
        chomp ($linebase);
         @dataline=split(/ /, $linebase);
         $qcand=$dataline[0];

        $rank=$dataline[3];
        $score=$dataline[4];
         if($qcand eq $q && $rank<=1000)
        {

             $doc=$dataline[2];
             $baselinescore{$doc}=$score;
      #print "query cand $qcand doc $doc score $score\n";

             if($maxbaselinescore<$score)
             {
             $maxbaselinescore=$score;
             }
               $conceptsdoclist=$docconcepts{$doc};

             @conceptsDoc=split(/ /,$conceptsdoclist);
             foreach my $concept (@conceptsDoc)
             {

                 $conctf=$tf{$concept.$doc};
                 $concn=$ndoc{$concept.$doc};
               #print "doc $doc conc $concept tf $conctf ndoc $concn\n";


                 if(exists($QConW{$concept}))
                 {
                      print "concept $concept exists in query and doc $doc\n";

$newscore{$doc}+=$conctf*log($Ntotaldoc/$concn)*$QConW{$concept};

                 }
                 $Normdoc{$doc}+=($conctf*log($Ntotaldoc/$concn))**2;
             } # end for each concept in doc

        }
    } # end each line in baseline

    $maxconScore=0;

    while (($doc,$score) = each(%newscore))
    {
        #$newscore{$doc}=$newscore{$doc}/(sqrt($Normdoc{$doc})*sqrt($NormQ));
        if($maxconScore<$newscore{$doc})
        {
            $maxconScore=$newscore{$doc};
        }
    }

      # COMBINING THE NORMALIZED SCORES TOGETHER
    while (($doc,$score) = each(%newscore))
    {
        $newscore{$doc}=$alpha*$newscore{$doc}/$maxconScore + (1-
$alpha)*$baselinescore{$doc}/$maxbaselinescore;
    }

  #-/////////////////////////////////////////////////////
                  # build the output
                print "reranking *************** \n";
                  $r=0;
                  @sortdoc = sort ( { $newscore{$b} <=> $newscore{$a} }
keys(%newscore) );

                  foreach $doc (@sortdoc) {

                print OUT "$q Q0 $doc $r $newscore{$doc} rrCosine-caption\n" ;
                $r++;
                last if($r==1001)
                }
}
 close(OUT);
```

A.6 Reading Reports in order to find the ICD-9 Codes

First step in the chapter 5 methodology is reading each report in order to find ICD-9 codes. This step has been done in this following program.

```
import java.io.BufferedReader;
import java.io.*
import java.util.HashMap;

public class ReadingReport {

       public static void main(String[] args) throws IOException {

              int g = 1;


              FileWriter vector101;
              BufferedWriter out;

              //OUTPUT/Users/Hoda /Documents/workspace/VisitID
              vector101 = new FileWriter("/Users/Hoda/Desktop/ICD.txt");
              out = new BufferedWriter(vector101);

              int j=0;
              while (j<101711)
              {


if((j==65563 || j==65591 || j==65603 || j==65821 || j==66303 || j==66576 ||
j==66835 || j==66896 || j==67147 || j==67148
                                  || j==67298 || j==67340 || j==67396 ||
j==67454 || j==67612 || j==67613 || j==67624 || j==67643 || j==67674 ))

                     {
                            j++;
                            out.write("\n");
                            continue;
                     }

                     String filename="";
                     if((j>16383 && j<65537))
                     {
                            filename= "/Users/Hoda/Documents/Thesis-
Data/TrecMed11/report" + j + ".xml";
                     }
                     else
                     {
                            filename= "/Users/Hoda/Documents/Thesis-
Data/test/report" + j + ".xml";
                     }
                     //System.out.println(j);

                     //INPUT 1 --->
                     FileInputStream fstream;
                     try {
```

```
                              fstream = new FileInputStream(filename);
                        } catch (FileNotFoundException e) {

                              System.out.print("j!=" + j + " && ");
                              j++;
                              continue;
                        }
                        // Get the object of DataInputStream
                        DataInputStream in = new DataInputStream(fstream);
                        BufferedReader br = new BufferedReader(new
InputStreamReader(in));
                        String strLine;

                        //Array was made here
                        int i=0;

                        while ((strLine = br.readLine())  !=  null     )
                        {

                              if (strLine.contains("checksum"))
                              {
                                    String [] checksum = strLine.split(">");
                                    String [] checksum1 = checksum[1].split("<");
                                    out.write(j + " " + checksum1[0] + " ");
                              }
                              if (strLine.contains("admit_diagnosis")) {
                                    String [] admit_diagnosis =
strLine.split(">");
                                    if(admit_diagnosis.length==1){
                                          out.write("XXX");
                                          i++;
                                          continue;
                                    }
                                    String [] admit_diagnosis1 =
admit_diagnosis[1].split("<");

                                    out.write(admit_diagnosis1[0] + ",");
                              }
                              if (strLine.contains("discharge_diagnosis")) {
                                    String [] discharge_diagnosis =
strLine.split(">");
                                    if(discharge_diagnosis.length==1){
                                          out.write("XXX");
                                          i++;
                                          continue;
                                    }
String [] discharge_diagnosis1 = discharge_diagnosis[1].split("<");

                                    out.write(discharge_diagnosis1[0]);
                              }

                              if (i > 10) {
                                    break;
                              }
                              i++;
                              out.flush();
                        }


                        out.flush();

                        j++;
                        out.write("\n");
                        in.close();
                  }

            }
}
```

A.7 Finding ICD-9 Codes Description

In order to find ICD-9 descriptions (Chapter 5) to proceed to find the weights for the terms and add them to the queries we run the following program to gather all the code's descriptions.

```
import java.io.BufferedReader;
       import java.io.BufferedWriter;
       import java.io.DataInputStream;
       import java.io.FileInputStream;

       import java.io.FileWriter;
       import java.io.IOException;
       import java.io.InputStreamReader;


       public class FindingCodesDescription {


              public static void main(String[] args) throws IOException
              {
                     //outputfile
                     FileWriter group = new
FileWriter("/Users/Hoda/Documents/query
expansion/TOPcodesDescriptionsTop50.txt");
                     BufferedWriter out = new BufferedWriter(group);

                     //inputfile1
                     FileInputStream fstream1 = new
FileInputStream("/Users/Hoda/Documents/query expansion/Topcodesforqueries-
togetthedescriptionTOP50.txt");
                     // Get the object of DataInputStream
                     DataInputStream in1 = new DataInputStream(fstream1);
                     BufferedReader br1 = new BufferedReader(new
InputStreamReader(in1));


                     String strLine;
                     String strLine2;

                     while((strLine = br1.readLine())!= null)
                     {
                            //inputfile2
                            FileInputStream fstream2 = new
FileInputStream("/Users/Hoda/Documents/query expansion/icd9-D+P.txt");
```

```
                            // Get the object of DataInputStream
                            DataInputStream in2 = new DataInputStream(fstream2);
                            BufferedReader br2 = new BufferedReader(new
InputStreamReader(in2));


                            String[] code = strLine.split(" ");


                            System.out.println(code[1]);
                            //System.exit(0);

                            while((strLine2 = br2.readLine())!= null)
                            {
                                   //System.out.println("strLine2 = " +
strLine2);
                                   String[] des = strLine2.split(";");
                                   if
(strLine2.toLowerCase().contains(code[1].toLowerCase())) {
                                          out.write(strLine + " " + des[1]+ "\n");
                                   }

                            }
                     }


                     out.flush();
                     System.out.println("hoda done UNIQUE1");
              }

       }
```

A.8 Weight for the ICD-9 Codes Descrptions for Each Query

Calculating each terms of codes descriptions by TF-IDF in the Chapter 5 has been done by the program that can we bring in this section.

```
public static void main(String[] args) throws IOException {
		double totalNumberOfReports = 17011.5;
		//OUTPUT
		FileWriter vector101;
		BufferedWriter out;
		vector101 = new
FileWriter("/Users/Hoda/Desktop/weightsforTop50.TXT");
		out= new BufferedWriter(vector101);
		out.write("<CONCEPT> <TF-IDF>\n");
		out.flush();


		//INPUT 1 --->
FileInputStream fstream = new
FileInputStream("/Users/Hoda/Downloads/TOPcodesDescriptionsTop50StemStop.txt");
		DataInputStream in = new DataInputStream(fstream);
		BufferedReader br = new BufferedReader(new InputStreamReader(in));
		String strLine;


		String prev = "101";
		out.write(prev);
		out.flush();
		//TODO File = INITIALIZE READ INPUT FOR TextSS
		while((strLine = br.readLine())!=null)
		{
			String[] stripped = strLine.split(" ");
			//System.out.println("stripped[0] = " + stripped[0]);
			//System.out.println("stripped.length = " +
stripped.length);

			if(!prev.equals(stripped[0]))
			{
				out.write("\n" + stripped[0]);
				prev = stripped[0];
			}


			for(int i=3;i<stripped.length;i++)
			{
				String word = stripped[i];
				//System.out.println("word = " + word);

				//INPUT 2 --->
				FileInputStream fstream2 = new
FileInputStream("/Users/Hoda/Documents/query expansion/Query" + stripped[0] +
"/"
						+ stripped[0] + "Top50ALLTerms.txt");
				DataInputStream in2 = new DataInputStream(fstream2);
```

```
                                        BufferedReader br2 = new BufferedReader(new
InputStreamReader(in2));
                                        String strLine2;

                                        double tfidfsum=0.0;
                                        while((strLine2 = br2.readLine())!=null)
                                        {
                                                String [] stripped2 = strLine2.split(" ");
                                                //System.out.println("stripped2[2] = " +
stripped2[2]);
                                                if(stripped2[2].equals(word))
                                                {
                                                        //System.out.println("stripped2[2] = " +
stripped2[2]);
                                                        //System.exit(0);
                                                        double TF =
Integer.parseInt(stripped2[3]);

                                                        String ntString = stripped2[5];
                                                        ntString = ntString.replaceFirst("Nt=",
"");
                                                        double ntInt =
Integer.parseInt(ntString);
                                                        double ntInt2= ntInt+0.5;
                                                        double IDF =
Math.log10(totalNumberOfReports/ntInt2);

                                                        double TFIDF = TF*IDF;

                                                        tfidfsum+=TFIDF;
                                                }
                                        }
                                        out.write(" " + word + "^" + (tfidfsum/20));

                                }
                                out.flush();
                        }
                }
}
```

A.9 Phyton Program for Stemming and Remove Stop Words

To add ICD-9 codes descriptions terms in chapter 5 we needed to remove stop words and stem the terms. This step has been done in the following program:

```
import re
from nltk.corpus import stopwords
from stemming.porter2 import stem

inFile = open('/home/ksz/Development/stemmer/input')

outFile = open('/home/ksz/Development/stemmer/textSS','w')
```

```
documents = inFile.readlines()
documents = ''.join(documents)
print ("₩n₩n")
#print(documents)
#print("₩n₩n")
#print("======================")
#print("₩n₩n")
#We only want to work with lowercase for the comparisons
documents = documents.lower()
#remove punctuation and split into seperate words
#words = re.findall(r'₩w+', documents,flags = re.UNICODE | re.LOCALE)
documents = documents.splitlines(True)

#This is the simple way to remove stop words
important_words=[]
i=0
for sentence in documents:
    for word in re.split(r'(₩W)', sentence):
        if i==1:
            i=0
            continue
        #bj = re.search(r"(₩w+)",word)
        #m_obj.group(1)
        if word not in stopwords.words('english'):
            important_words.append(word)
        else:
            i=1

#print ("₩n₩n")
#print(important_words)
#print("₩n₩n")
print("======================")
#print("₩n₩n")
#documents = [[stem(word) for word in sentence.split(" ")] for sentence in documents]
documents = [stem(word) for word in important_words]
#print ("₩n₩n")
for word in documents:
    outFile.write(word)
```

```
#if(word.count("₩n")>0):
    #out.write(word)
    ##print(word, end="")
#else:
    #out.write(word)
    #out.write(" ")
    ##print(word,end=" ")
```

B. Topics

B.1 Topics of Medical TREC 2011

101 Patients with hearing loss
102 Patients with complicated GERD who receive endoscopy
103 Hospitalized patients treated for methicillin-resistant Staphylococcus aureus (MRSA) endocarditis
104 Patients diagnosed with localized prostate cancer and treated with robotic surgery
105 Patients with dementia
106 Patients who had positron emission tomography (PET), magnetic resonance imaging (MRI), or computed tomography (CT) for staging or monitoring of cancer
107 Patients with ductal carcinoma in situ (DCIS)
108 Patients treated for vascular claudication surgically
109 Women with osteopenia
110 Patients being discharged from the hospital on

hemodialysis

111 Patients with chronic back pain who receive an intraspinal pain-medicine pump

112 Female patients with breast cancer with mastectomies during admission

113 Adult patients who received colonoscopies during admission which revealed adenocarcinoma

114 Adult patients discharged home with palliative care / home hospice

115 Adult patients who are admitted with an asthma exacerbation

116 Patients who received methotrexate for cancer treatment while in the hospital

117 Patients with Post-traumatic Stress Disorder

118 Adults who received a coronary stent during an admission

119 Adult patients who presented to the emergency room with with anion gap acidosis secondary to insulin dependent diabetes

120 Patients admitted for treatment of CHF exacerbation

121 Patients with CAD who presented to the Emergency Department with Acute Coronary Syndrome and were given Plavix

122 Patients who received total parenteral nutrition while in the hospital

123 Diabetic patients who received diabetic education in the hospital

124 Patients who present to the hospital with episodes of acute loss of vision secondary to glaucoma

125 Patients co-infected with Hepatitis C and HIV

126 Patients admitted with a diagnosis of multiple sclerosis

127 Patients admitted with morbid obesity and secondary diseases of diabetes and or hypertension

128 Patients admitted for hip or knee surgery who were treated with anti-coagulant medications post-op
129 Patients admitted with chest pain and assessed with CT angiography
131 Patients who underwent minimally invasive abdominal surgery
132 Patients admitted for surgery of the cervical spine for fusion or discectomy
133 Patients admitted for care who take herbal products for osteoarthritis
134 Patients admitted with chronic seizure disorder to control seizure activity
135 Cancer patients with liver metastasis treated in the hospital who underwent a procedure

B.2 Example for Added ICD-9 Codes to the Queries

This appendix includes the first five topics with their most frequent ICD-9 codes for Top 50 retrieved reports (chapter 5). The columns stand for number of query, ICD-9 code, number of frequency and code's description.

101	389.9	9	Unspecified hearing loss
101	401.9	5	Unspecified
101	428.0	4	Congestive heart failure unspecified
101	414.01	4	Of native coronary artery
102	401.9	8	Unspecified
102	530.81	7	Esophageal reflux
103	790.7	6	Bacteremia
103	401.9	6	Unspecified
103	311	6	Depressive disorder not elsewhere classified
103	041.11	6	Methicillin susceptible Staphylococcus aureus

103	276.1	5	Hyposmolality and/or hyponatremia
103	530.81	4	Esophageal reflux
103	038.11	4	Methicillin susceptible Staphylococcus aureus septicemia
104	401.9	13	Unspecified
104	530.81	6	Esophageal reflux
104	414.01	5	Of native coronary artery
105	401.9	11	Unspecified
105	294.8	10	Other persistent mental disorders due to conditions classified elsewhere
105	331.0	7	Alzheimer's disease