Spring 1-1-2012

A Comparison of Lexical Expansion Methodologies to Improve Medical Question and Answering Systems

Timothy Wilson St. Charles
University of Colorado at Boulder, stcharle@colorado.edu

A Comparison of Lexical Expansion Methodologies to Improve Medical Question and Answering Systems

by

Mr. Wil St. Charles

B.A., University of Colorado, Boulder, 2010

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Masters of Science

Department of Computer Science

2012

This thesis entitled:
A Comparison of Lexical Expansion Methodologies to Improve Medical Question and Answering Systems
written by Mr. Wil St. Charles
has been approved for the Department of Computer Science

James Martin

Prof. Wayne Ward

Ph.D. Rodney Nielsen

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Wil St. Charles, Mr. (M.S. Computer Science)

A Comparison of Lexical Expansion Methodologies to Improve Medical Question and Answering Systems

Thesis directed by Prof. James Martin

In this paper, a variety of lexical expansion approaches were evaluated using the Medpedia corpus and MiPACQ queries in order to improve the MiPACQ system's retrieval performance. The heart of the MiPACQ system is a document reranking component, and this component utilizes the results from a baseline information retrieval system. However, the baseline IR system used in MiPACQ has poor paragraph level *recall* performance which limits the reranker's overall performance. To help solve these issues, three broad term expansion approaches are outlined in this paper with the purpose of increasing *recall* over the baseline Lucene retrieval system without introducing a significant amount of noise. Two of the three expansion approaches only rely on the corpus being indexed, while the last expansion technique requires a domain specific ontology to expand query terms. First, automatic thesaurus generation based on co-occurrences is evaluated as an expansion methodology along side other co-occurrence based expansion methods. Next, a resource based approach that uses the UMLS Metathesaurus for expansion is used to evaluate knowledge rich expansion methods. Finally, latent semantic indexing is evaluated as an alternative to the baseline vector space retrieval model. These methods are compared and tweaked and the best method is recommended to the MiPACQ authors to improve Q & A results.

Dedication

To my loving parents.

Acknowledgements

Thank you to Prof. Martin for advising this thesis and helping me with my many questions along the way. Thank you to Rodney Nielsen for providing me with queries, relevance judgments, and the Medpedia document dumps.

Contents

Tables

Table

Figures

Figure

Chapter 1

Introduction

The medical field is generating an enormous amount of data that is outstripping the traditional methods for accessing and utilizing this data. Many clinical physicians are unable to fully take advantage of the large amount of medical data that is becoming available to them. Medical information retrieval systems attempt to address this problem by providing a framework for searching large amounts of medical free text. Traditional medical IR systems focus on returning relevant documents to a user's free text queries; however, documents are often quite long and it can take an unacceptably long time for a physician to determine if her information need has been fulfilled by a document. More recent work has started to focus on medical question and answering (Q & A) systems, which are geared towards returning a piece of text or a document that is targeted at quickly fulfilling a user's information need. To achieve this goal, medical Q & A systems apply a large range of natural language processing techniques (NLP) to get a better understanding of the user's query and the information need it describes.

MiPACQ is a recently developed medical Q & A system built as a NLP pipeline on top of a standard information retrieval system (Lucene's vector space retrieval model). The pipeline consists of a few separate stages: annotating the question with its expected answer type; semantically labeling the question; retrieving the top n documents from the baseline IR system; semantically annotating the result set; and reranking the documents retrieved by the standard IR system. The document reranker is responsible for actually determining the final order of documents that get returned, so it directly impacts the system's overall retrieval performance. However, the reranker can only rerank documents that have been returned the by standard IR system. This does not cause any problems when the standard IR system's *recall* performance

is sufficiently high; however, the baseline retrieval performance is often not sufficient for many medical Q & A systems such as MiPACQ.

The MiPACQ Medpedia Q & A corpus is created by taking large documents and segmenting them into sections, usually paragraphs, to index and return to the user. These small paragraphs contain a limited set of terms compared to the documents from which they are derived, and these types of paragraphs yield low *recall* performance when evaluated using the short to medium length free text queries in the MiPACQ evaluation set. Figure 1.1 shows the average overlap percentage between each MiPACQ query and its relevant Medpedia paragraphs, while figure 1.2 shows the same information for MiPACQ queries and their relevant Medpedia documents. As seen in the figures, MiPACQ queries and Medpedia paragraphs have significantly less term overlap than MiPACQ queries and Medpedia documents. Low query term and document term overlap causes problems for the vector space retrieval model which relies on term overlap to help determine document relevance and ranking. Often this problem manifests itself in retrieval results as poor *recall* performance, and this is the case for the standard IR system present in the MiPACQ system.

Query term and document term mismatch is a problem that has been addressed in the information retrieval field by a variety of query and document expansion methods. The general concept behind query expansion methods is to expand query terms with synonyms to increase *recall* performance. Additional query terms increase the likelihood that the user's query matches a document, improving retrieval performance. There are a variety of popular methods to implement query expansion that require a varying degree of user interaction and outside knowledge. This paper examines three broad approaches to query expansion: automatic thesaurus generation and expansion; resource based query expansion; and latent semantic indexing. These approaches are evaluated by examining the full set of documents returned by the IR system with a specific emphasis placed on *recall* performance. [1] This evaluation method allows for a direct comparison of the real *recall* gains made by each system, and by not truncating the results at the top n documents, *recall* scores are unaffected by precision gains. Precision is measured by the mean average precision (MAP), mean reciprocal rank (MRR), and precision at one ($P@1$) statistics that are collected for each system.

[1] The MiPACQ system described in the paper only reranks the top N documents returned by the baseline IR system [9].

Figure 1.1: Query term and paragraph term overlap in the Medpedia corpus and MiPACQ query set

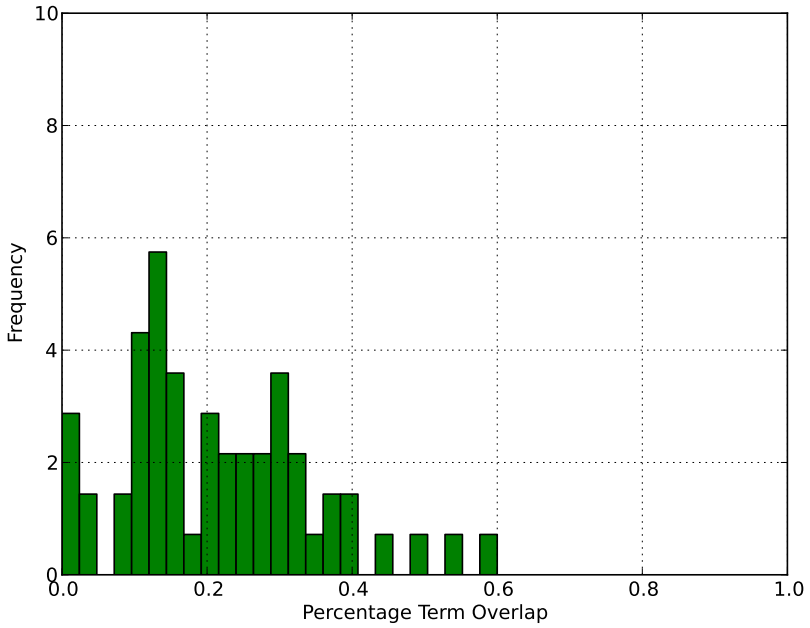
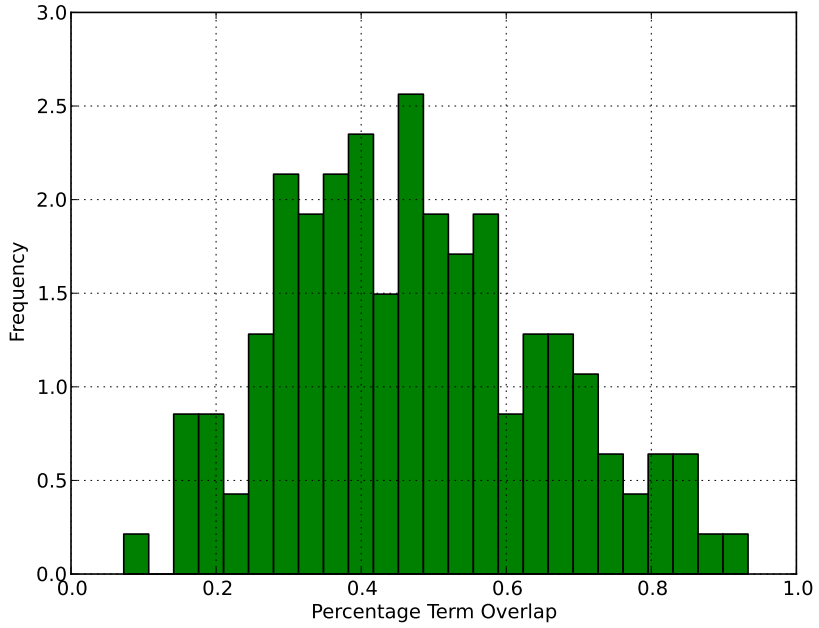


Figure 1.2: Query term and document term overlap in the Medpedia corpus and MiPACQ query set



Chapter 2

Prior Work

Query and document expansion has been reasonably well studied for general information retrieval systems. Bhogal, Macfarlane, and Smith define the immediate goal of query expansion "is to add new meaningful terms to the initial query" [6]. The definition of "meaningful terms" varies, but generally they provide additional context for the original query terms. In addition to providing context, there are two types of problems that query expansion helps to address in information retrieval. Synonymy causes query term and document term mismatch because two distinct terms can refer to the same thing. Granularity can cause mismatch when the query term and document term describe an object at two separate levels of granularity (e.g. disease vs. Polio). These types of term mismatch lead to poor retrieval performance in the vector space model, but these drawbacks can be mitigated by introducing new terms from a query expansion system [15].

The generation of expansion terms can be done in an automatic, semi-automatic, or manual fashion, but this paper will only examine automatic query expansion methods. Automatic expansion occurs after a user has submitted a query and does not require any additional input from the user. This type of expansion is achieved by one of a few types of major systems: ontology based expansion systems; co-occurrence based expansion systems; and relevance feedback systems. Given the right set of parameters, all of these approaches can systematically improve retrieval performance [6].

One well studied approach to query expansion uses a human generated ontology to expand queries at runtime. Ontologies have a myriad of definitions, but they all "provide consistent vocabularies and world representations necessary for clear communication within knowledge domains" [6]. The predominate ontology used in the medical domain is the UMLS Metathesaurus (section 3.3), which was created to link

different medical vocabularies together, but it also provides an extensive set of relations between medical terms [21]. In most ontologies, including the UMLS Metathesaurus, there are a few broad categories of relationships expressed between entities: synonym; hierarchical; and related. Synonym relationships denote equivalence while hierarchical relationships indicate relations like "is a part of" and "is made up of", and related relationships cover everything else (e.g. "symptom of" or "treatment for") [15]. Most ontology based query expansion strategies use some form of this taxonomic information to do query expansion (e.g. synonyms or hypernyms) [20]. Work done by Hersh, Price, and Donohoe shows that query expansion using the synonym relationship provided the best retrieval results, but it did not show a significant aggregate boost of retrieval performance on their data set [15]. However, their data set had already been labeled with MeSH terms, which have been shown to provide a significant positive boost to retrieval performance. It is likely that an unlabeled free text corpus would show greater retrieval gains from this type of expansion.

Another study that used the UMLS Metathesaurus in conjunction with the UMLS Metamap program showed favorable retrieval gains compared to a relevance feedback system [4]. This system uses the UMLS Metamap NLP pipeline to identify UMLS entity occurrences in free text queries. However, instead of using the synonym relationship to get expanded terms, the system uses the canonical name of the entity identified by the UMLS Metamap tool. The authors suggest that the canonical name is a good candidate for expansion because it often matches human assigned MeSH terms included in the indexed document. Additionally, the system uses a weighting scheme to weight the canonical entity names over twice as high as the original term occurrences. Using both the human assigned MeSH terms and the Metamap query expansion, the expansion system showed a 16% improvement in average precision over the baseline system [4].

Depending on the exact corpus and expansion methodology, UMLS Metathesaurus based expansion can have a positive impact on retrieval performance. Additionally, a knowledge based expansion approach using the UMLS is attractive because it avoids some of the traditional problems with knowledge source based expansion. In many knowledge source based query expansion approaches, word sense ambiguity is a large problem when expanding queries; however, in the UMLS Metathesaurus less than 1% of terms are assigned multiple concepts. This lack of ambiguity side steps one of the largest problems associated with query expansion, and helps avoid the loss of precision that can be caused by query expansion. Some studies also

suggest that expanding both documents and queries with more general and more specific UMLS concepts respectively can effective address granularity mismatch problems in the medical domain [16].

However, there is evidence to suggest that query expansion strategies that use co-occurring terms perform better than ones that use taxonomic information. Co-occurring terms are defined by Navigli and Verladi as "words that, on a probabilistic ground, are believed to pertain to the same semantic domain (e.g. car and driver)" [20]. In practice co-occurring terms are determined to be semantically related if they occur in the same document together. These co-occurrence statistics are used to generate global thesauri which in turn are used for query expansion. Thesauri are traditionally used in query expansion to enhance recall by broadening the query with additional terms, which is particularly useful when the output of the retrieval system is reranked like in modern Q & A systems. Global thesauri are generated using the co-occurrence statistics from the entire unmodified corpus. This is opposed to local thesauri which are generated at query time by only using co-occurrence statistics from documents returned by querying the corpus with an unmodified query [13]. Although global thesauri share a name with traditional thesauri, they are not made up of synonyms in the traditional sense. Instead global thesauri consist of closely related terms in the context of the corpus, so often thesauri consist of semantically related terms but not classic synonyms [13].

One of the simplest approaches to creating a global thesaurus is to calculate term-term similarities [18, p192-193]. Term-term similarities are calculated by taking a term by document matrix and multiplying it with its transpose; the resultant matrix is a term by term matrix whose entries represent term similarity with larger values representing stronger relationships. This method has the advantage of being straight forward to implement and performant on large document collections. However, as with most thesaurus generation techniques, it does not disambiguate word senses, and this can have a negative impact on retrieval performance for many applications [18, p192-193]. As covered earlier, this is less of a problem in the medical domain due to many medical terms only having one word sense.

Another approach to generating thesauri is to cluster documents, and create thesaurus classes from the lower frequency terms in each document cluster. This approach requires a specific type of clustering to generate meaningful thesaurus classes. Clusters should be small and very closely related if the low frequency terms are going to be put in a thesaurus class together. Complete link agglomerative clustering is a common

choice to produce these types of clusters [13]. Cluster based thesaurus generation and expansion has been shown to provide small gains to average precision over a variety of corpora [12].

Query expansion is not the only way to address synonymy and granularity problems in information retrieval and Q & A systems. Document expansion is an alternative to query expansion that functions in the same way, but instead of expanding queries at runtime the system expands documents at index time. Document expansion is typically performed by the same systems that do query expansion by treating documents as long queries and indexing the expanded output. Ideally, this type of expansion provides the same benefits that query expansion does, but with the benefit of reduced runtime costs at query time [8].

In general information retrieval tasks document expansion using co-occurrence statistics has been shown to provide small gains over a baseline system without expansion, but it falls short of reaching the retrieval performance gains of traditional query expansion systems [8]. However, these results are based on local expansion techniques which may fare worse on document expansion tasks than query expansion tasks. Other document expansion systems use outside knowledge sources (ontologies) to expand documents. Generally, these systems perform similarly to the ontology based query expansion systems described earlier. However, some interesting work has been done on specific methods for utilizing ontologies for document expansion. Agirre et al. describe a method that uses random walks through WordNet to label documents with concepts that represent the document as a whole [1]. These random walks are implemented via a personalized PageRank algorithm, where concepts relevant to the document get assigned a high PageRank score. This approach showed retrieval gains across three data sets made up of disparate document types.

Latent semantic indexing (LSI) is an indexing approach related to document expansion in that it tries to solve the problems of synonymy and granularity by modifying the document collection. The aim of LSI is to "construct a "semantic" space wherein terms and documents closely associated are placed near one another" [14]. This semantic space is usually significantly smaller than the original term-document space, and previously syntactically distinct but semantically related terms can now be represented by a single dimension. The original paper on LSI showed that LSI provided significant retrieval performance gains on the MED corpus [14]. Additionally, on the more modern TREC retrieval tasks LSI has been shown to provide retrieval gains of up to 15% [5]. However, patient record retrieval using concept based LSI did not

yield comparable results to the original gains on the MED corpus [10]. Still, latent semantic indexing can potentially be used to boost retrieval performance in medical Q & A systems.

A large portion of the work done on query expansion shows that it is effective at increasing *recall* performance. This is the metric which the systems analyzed in this paper are measured against because these systems are being evaluated in the context of full medical Q & A systems. As discussed earlier, the MiPACQ medical Q & A system relies on a reranker to provide retrieval gains, but even perfect reranker is limited by the documents it is supplied to rerank. If documents are missing from the set returned by the underlying information retrieval system then overall system performance will suffer. Query and document expansion systems are ideal to fix this problem.

Chapter 3

Test Environment and Setup

3.1 Lucene

Apache Lucene is an open source information retrieval system implemented in Java. Lucene is an implementation of the vector space information retrieval model. Documents and queries are represented as vectors in term space, and documents are ranked and returned by their closeness to the query vector. Lucene serves as the foundation for the baseline Q & A system and as the starting point for many of the query expansion experiments.

3.1.1 PyLucene Python Wrapper for Lucene

PyLucene is a set of Python bindings for Lucene maintained by the Apache Software Foundation. These bindings expose all of Lucene's functionality to Python programs. Under the hood, the PyLucene wrappers use JCC to generate C++ code that calls into the Java Lucene implementation.

3.2 LexEVS Terminology Server

LexEVS is a terminology server developed by the National Cancer Institute and the Mayo Clinic. It supports loading and serving the full UMLS Metathesaurus as well as specific UMLS subsets. LexEVS is written in Java and provides programmatic access to a large variety of ontology formats. Additionally, the LexEVS server can be hosted in a Java Applet server (Tomcat or Glassfish) to provide access to the stored terminology through RESTFUL queries.

3.2.1 RESTFUL Query Interface

The LexEVS server functionality can be exposed via a Java Web Applet running in a standard Java Servlet Container (e.g. Tomcat or Glassfish). This applet exposes both a web based interface and a RESTFUL API. The RESTFUL API exposes the core LexEVS functionality, such as retrieving relationships between entities and finding the canonical name of an entity. The LexEVS applet returns the information from an API request as XML.

3.3 UMLS Metathesaurus

The Unified Medical Language System (UMLS) is a collection of controlled medical ontologies packaged together in a common format. It not only provides a way to map concepts from one ontology to another, but it also serves as the most available and complete thesaurus and ontology of medical terms. The UMLS Metathesaurus is maintained by and freely available from the US National Library of Medicine.

3.3.1 Metamap

Metamap is a free text processing system that maps free text phrases to UMLS concepts. It consists of a reasonably complete Natural Language Processing pipeline that consists of a few basic operations: tokenization; abbreviation expansion; part of speech tagging; lexical look up using the SPECIALIST lexicon; variant generation; candidate identification and mapping; and word sense disambiguation [3].

3.3.2 MetamorphoSys

MetamorphoSys is a tool packaged with the UMLS Metathesaurus that allows users to add and remove ontologies from the Metathesaurus. This tool takes the UMLS Metathesaurus as input and outputs a subset of the ontologies included in the Metathesaurus.

3.3.3 SNOMED-CT

Systematized Nomenclature of Medicine - Clinical Terms (SNOMED-CT) is one of the medical ontologies included in the UMLS Metathesaurus. Its entities predominately fall under one of a few categories:

diseases, findings, procedures, organisms, and anatomy. Due to its wide range of topics and relatively small size compared to the full Metathesaurus, SNOMED-CT is a good candidate for testing many resource based query expansion techniques.

3.4 Python Numerical Libraries

The standard Python distribution does not contain built in support for scientific computing; however, there are a number of Python packages that add this functionality.

3.4.1 Numpy

Numeric Python (Numpy) is a package for Python that adds arrays and many numeric primitives to Python. Additionally, it adds support for a variety of array operations via broadcasting, and the package includes support for useful linear algebra and random number generation tools. It is open source and is maintained by the Numpy Developers Group.

3.4.2 Scipy

Scientific Python (Scipy) is an open source Python package built on top of Numpy for scientific computing. It contains many useful modules for IR such as sparse arrays and advanced linear algebra (Decompositions, Spatial Distance, etc).

3.4.2.1 Scipy.sparse

Scipy's sparse array package allows users to build memory efficient sparse arrays and matrices (e.g. matrices that are predominately filled with zeros). Additionally, Scipy.sparse supports a variety of sparse array formats and supports easy and efficient conversion between supported formats. This allows users to build sparse arrays using an efficient format for creation and then convert them to an efficient format for manipulation (searching). These sparse matrices serve as the basis for the index in the latent semantic indexing section, and they also serve as the format for term by document matrix used to construct the global thesauri.

3.4.3 SparseSVD

SparseSVD is a Python package which is a simple wrapper around SVDLIBC (a C library for sparse singular value decompositions). This package provides a convenient and memory efficient way to calculate fairly large singular value decompositions of sparse matrices. It takes a Scipy.sparse sparse matrix as input and outputs the decomposition as three sparse matrices (U, S, VT).

3.5 Medpedia

Medpedia is a comprehensive collaborative encyclopedia that is built and maintained by verified medical professionals. It is based on the Wikipedia model of information sharing, and allows any verified medical professional to contribute articles to the knowledge base. However, anyone from the public has free access to the information contained in Medpedia. Even though Medpedia is still early in its life cycle it contains a significant number of topics and articles. When the offline Medpedia dump that this paper uses was collected it contained roughly 8,000 articles and 600,000 paragraphs. This knowledge source serves as the corpus for the systems in this paper and the MiPACQ Q & A system.

3.6 Custom Number Analyzer

All of the systems in this paper that are built on top of Lucene use a custom number analyzer to tokenize and filter free text. This analyzer incorporates all the features of Lucene's standard analyzer such as stemming and stop wording, but it also filters out numbers from the free text token stream. The text to be indexed is fed into Lucene's standard tokenizer which tokenizes the input by white space and punctuation. It is then passed to Lucene's Standard Filter and Lowercase Filter, and finally it is passed through a custom stop filter and a custom number filter. The stop filter removes instances of the top fifty most common non-medical words in the Medpedia corpus, while the number filter excludes any numbers (both decimals and integers). In this context, a number is defined by the following regular expression.

$$[-+]?[0\text{-}9]*.?[0\text{-}9]+ \tag{3.1}$$

This filtering is performed to help trim the corpus down and eliminate noise because numbers with one occurrence make up almost a fourth of all unique terms in the Medpedia corpus. One advantage of this filtering is that it cuts down on the space required to store an index because the number of unique terms is significantly reduced with this analyzer. This has the side effect of also speeding up a number of calculations that depend on the term by document matrix.

3.7 Baseline Lucene Retrieval System

3.7.1 Baseline Document System Without Lexical Expansion

The baseline system uses PyLucene as the underlying search engine, and the baseline system indexes and searches the documents using a typical Lucene work flow. All of the relevance judgments, documents, and queries are stored in plain text files and parsed with regular expressions. The Medpedia corpus was retrieved for the original MiPACQ paper [9], and the corpus sans Wikipedia markup and formatting is stored in paragraph form in a flat text file. To create an indexable document, the system concatenates all the paragraphs that share a document ID and stores them as a single field in a document created with an IndexWriter. Each document is transformed from free text to an indexable set of tokens by the number analyzer (section 3.6). Once the text has been analyzed it is added to an index which is stored on disk using Java's New I/O libraries (Lucene NIOFS Directory). The baseline system is evaluated by running each query string through Lucene's built in Query Parser that has been initialized with the number analyzer described above. These returned results are then compared to the document level annotations to determine if they are relevant. Documents are either considered relevant or irrelevant, and the system does not take levels of relevance into account. This information is used to calculate the precision at one (P@1), *recall*, Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR) metrics.

3.7.2 Baseline Paragraph System Without Lexical Expansion

The second baseline system that is used through this paper is the baseline paragraph system. This system is almost identical to the baseline document system. However, instead of indexing entire documents it indexes individual paragraphs. Individual paragraphs are created from whole documents by splitting

the document into sections according to white space rules (line breaks). There are two sets of relevance judgments; one for evaluating document level relevance and one for evaluation paragraph level relevance. This system uses paragraph specific relevance judgments when calculating $P@1$, *recall*, MAP, and MRR. Other than the difference in relevance judgments, this system is identical to the baseline document system.

3.7.3 Evaluation of Ranked Results

The baseline paragraph and document systems return documents in a ranked order, as does every other system evaluated in this paper. These rankings are evaluated using four separate metrics: precision at one ($P@1$), *recall*, Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR). Each of these metrics relies on an annotated result set to determine if a particular paragraph is relevant. The MiPACQ paragraph level annotations have five annotation types: answer, partial answer, optional information, context, or irrelevant. Paragraphs annotated with the "answer" annotation type are treated as relevant when calculating all of the metrics. However, paragraphs annotated with the "partial answer", "optional", or "context" annotation types are treated as if they do not exist when calculating $P@1$, MAP, MRR, and *recall*. This is because paragraphs that are annotated with one of these types should not help or hurt a metric because they are neither entirely relevant or irrelevant. Paragraphs annotated with the "irrelevant" annotation type are treated the same as paragraphs that have not been annotated, and they are counted as irrelevant when calculating all of the metrics.

System	P@1	MAP	MRR	Overall Recall
Baseline Document System	0.6691	0.4251	0.7457	0.9624
Baseline Paragraph System	0.1379	0.0623	0.2265	0.6533

Table 3.1: Baseline Retrieval System Results

Chapter 4

Automatic Methods for Query and Document Expansion

4.1 Document Expansion at Index Time

System	P@1	MAP	MRR	Overall Recall
Baseline Paragraph System	0.1379	0.0623	0.2265	0.6533
Paragraph System with Top Term Expansion (25 Terms)	0.1552	0.0604	0.2334	0.6878
Paragraph System with Top Term Expansion (50 Terms)	0.1379	0.0590	0.2246	0.7314
Paragraph System with Top Term Expansion (100 Terms)	0.1035	0.0524	0.1930	0.7840
Paragraph System with Top Term Expansion (200 Terms)	0.1035	0.0557	0.1937	0.8858

Table 4.1: Document Expansion at Index Time Results

One potential way to address document term query term mismatch is to expand documents at index time. Index time expansion occurs by preprocessing the document before indexing to generate additional terms to index alongside the original text. These additional terms largely serve as synonyms for some semantic content present in the document, but these terms also can be used to add contextual information to narrowly focused documents. Document expansion at index time has a significant runtime overhead advantage compared to traditional query-time query expansion methods. Because document expansion is only done once for a document at index time, no additional runtime overhead is incurred at search time like there is with traditional query expansion methods.

4.1.1 Expanding Paragraphs by Using the Top Terms from their Parent Document

One of the simplest and most effective index time expansion methods is to expand paragraphs with the top N terms from their parent document. A paragraph is defined as a non-overlapping, ordered, continuous subset of terms in a document (e.g. standard English paragraphs). In the Medpedia Corpus a single document is split into seventy nine paragraphs on average, and due to this very fine partitioning each paragraph has limited contextual information embedded within it. To help contextualize each paragraph, this system indexes the top N terms from a paragraph's parent document alongside each paragraph. The top N terms are selected by taking the N terms with the highest $tf - idf$ weight from the parent document. This method leverages the baseline document system's very good *recall* performance to help improve the paragraph system's comparatively poor *recall* performance. This type of expansion has similar performance to query expansion using an automatic thesaurus generated with document co-occurrences (subsection 4.2.5). The similar performance of both systems that use document level information suggests that the context introduced by including document level information is good way to improve retrieval performance.

To calculate the top N terms from a paragraph's parent document, the system constructs a term by document matrix based on the documents from the Medpedia corpus. The term by document matrix is constructed in the same manner as described in 4.2.1. Each column in the term by document matrix represents a parent document, and the terms in each column are sorted by their $tf - idf$ weight. Once the term by document matrix is constructed, the system starts to index the paragraphs with the standard Lucene pipeline. However, before each paragraph is indexed via the IndexWriter, it is expanded in a pre-processing step. In the preprocessing step, the column in the term by document matrix that corresponds to the paragraph's parent document is used to retrieve the top N terms. These terms are then sent to the IndexWriter as a separate field to index from the main paragraph contents. Once the indexing is complete, the rest of the system runs identically to the baseline system with one caveat: when queries are run against this index, they are constructed to search both fields as opposed to the single original text field.

Table 4.1 shows the set of results from the top term expansion method tested with a few different parameters for N. This expansion method shows a solid improvement in *recall* over the baseline system,

and when $N = 50$ there is only a very minor drop in MRR and MAP. These results show that this type of expansion is an good candidate for improving MiPACQ results, because the first relevant document is roughly the same rank as it is in the baseline system and significantly more relevant paragraphs are included in the result set. Including document context comes at a price though as $P@1$ performance is significantly decreased for larger values of N when compared to the baseline system. However, this issue should largely be mitigated when the result set is reranked, and as such it should not a major detriment to performance.

4.2 Automatic Global Thesaurus Generation using Medpedia Corpus

The term co-occurrences in the Medpedia corpus can be used to construct a global thesaurus that can be used for query expansion. The premise behind this kind of thesaurus generation is that the more often two words co-occur together the more likely it is that they are semantically related to one another in the context of the corpus. This method has the advantage of being completely self contained, and it does not need any resources outside of the document collection being used to generate the thesaurus. However, there are two potential disadvantages: this type of system assumes that words that co-occur in the same document together are strongly semantically related and this might not be true; it can also be difficult to pick the correct parameters for automatic thesaurus generation.

This system uses one of the simplest methods for determining when two terms co-occur. If two terms occur in the same document or paragraph, that is considered a co-occurrence. The weight of a co-occurrence is the product of the weight of each term in the term by document matrix (usually a $tf - idf$ weight). This simple method for determining co-occurrences was chosen for its speed and straight forward implementation.

4.2.1 Constructing the Term by Document Matrix

The term by document matrix is constructed from an existing Lucene index that has been generated using the standard Lucene work flow including the number analyzer described in 3.6.

The Lucene index is opened via an $IndexReader$, and the sparse matrix is allocated with dimensions retrieved from the $terms()$ and $numDocs()$ methods of the $IndexReader$ which correspond to the number of unique terms in the index and the total number of documents in the index. Once the sparse matrix is

allocated, it is filled in by iterating through all of the terms in the index. Each matrix entry is filled in according to whether the system is using tf-idf, term count, TF Normalization, or cosine normalization. The simplest method is to use raw term frequency as is described below, but the other methods usually yield better results as described in 4.2.3.

To populate each entry in the term by document matrix with term frequency, the system iterates through a *termDoc* object for each term in the Lucene index. For a given term, a *termDoc* object provides a method for iterating through $< document, frequency >$ pairs. The system iterates through every term and retrieves the corresponding *termDoc* object as well as storing a term text to term ID (row number) mapping. Using the retrieved *termDoc* object, the system loops over each $< document, frequency >$ pair filling in the appropriate term document entry in the matrix.

Once the term by document matrix is populated, it is converted from the initial list-of-lists sparse matrix format to a compressed sparse column format. The compressed sparse column format allows for more efficient manipulation of the matrix when performing the matrix multiplication to compute the term co-occurrences. Even for the moderately size Medpedia corpus, the term by document matrix is small enough to fit in the RAM of our test system (4 GB).

4.2.2 Creating the Co-occurrence Matrix

The co-occurrence matrix is created by multiplying the term by document matrix by its transpose (equation 4.1), yielding a term by term matrix. Each entry in the co-occurrence matrix is a score that is representative of relatedness of two terms.

$$M_{i,j} = \sum_{k=1}^{\text{Total terms}} D_{i,k} * D_{k,j}^{T} \tag{4.1}$$

Equation 4.1 describes how the co-occurrence matrix is filled out where M is the co-occurrence matrix and D is the term by document matrix. Each entry $D_{i,j}$ is filled with with a term frequency or $tf - idf$ value, so the resulting entry Mi, j is a value that represents how significant the relationship is between $term_i$ and $term_j$. Instead of comparing these co-occurrence values in absolute terms, the system normalizes each term row by the maximum co-occurrence value in that row (equations 4.2 through 4.4).

$$\vec{T_i} =< M_{i,0}, M_{i,1}, ..., M_{i,N-1}, M_{i,N} > \tag{4.2}$$

$$\vec{T_i} = \frac{\vec{T_i}}{Max(|\vec{T_i}|)} \tag{4.3}$$

$$M_{i,*} = \vec{T_i} \tag{4.4}$$

This makes the co-occurrences of every term directly comparable, where a perfectly correlated pair of terms has a value of 1.0. A benefit of this normalization is that co-occurrences of common terms and rare terms can be directly compared. Additionally, this normalization allows the user to specify an intuitive cutoff value between 0.0 and 1.0 to control the number of synonyms included in the thesaurus. This cutoff value represents how close a term has to be to the best co-occurrence value to be included in the thesaurus. The best co-occurrence value for any given term is almost always the term with itself (the identity pairing), so a co-occurrence value of .5 implies that to make it into the thesaurus a co-occurrence has to exist in at least half as many documents as the term appears in. The effects of changing the cutoff value are explored more in section 4.2.4

Due to the size of the corpus, the term by document multiplication described in equation 4.1 can be quite resource intensive. This system utilizes a block matrix multiply that clamps the result of the block multiply by the cutoff value before moving on to the next block. The end result is identical to performing a standard matrix multiplication and then an independent clamping pass; however, this approach is significantly more memory efficient as only the intermediate block result needs to be stored in memory. This resulted in significant memory savings, and allowed the entire matrix multiplication to be performed in RAM on a 4 GB system with the Medpedia corpus.

4.2.3 TF-IDF, TF Normalization, and Cosine Normalization

There are several modifications to the construction of the term by document matrix that can improve the results produced by the automatic thesaurus generation query expansion system.

4.2.3.1 TF-IDF

The basic term by document matrix is constructed using pure term frequencies. This has a number of disadvantages for automatic thesaurus generation; the primary disadvantage is that it weights occurrences of terms linearly. This weighting means that if a term co-occurs with another term ten times then that relationship is twice as important as a similar pairing that only occurs five times. Additionally, pure term frequencies do not take into account the rarity of a word which undervalues co-occurrences that include words that occur with a low base rate. This happens because terms with low base rates have smaller term frequencies, so when the product of term frequencies is used to measure relatedness it undervalues terms with a low base rate of occurrence because they yield smaller term frequency products. Both of these problems can be fixed by using $tf - idf$ weighting when constructing the term by document matrix.

$$T_f = 1 + \lg(\text{Raw term frequency}) \tag{4.5}$$

$$IDF(t_i) = 1 + \lg\left(\frac{\text{Total number of documents}}{\text{Number of documents } t_i \text{ occurs in } + 1}\right) \tag{4.6}$$

Instead of using raw term frequencies, the $tf - idf$ weighting scheme uses equation 4.5 to dampen the effect of large term counts. Likewise, the IDF equation (4.6) helps to account for term rarity when computing co-occurrences. This weighting scheme only requires small modifications to how the term by document matrix is constructed, and it yields minor improvements to retrieval performance.

Table 4.2 shows the *recall*, MAP, and MRR improvements gained by moving from raw term frequencies to $tf - idf$ weighting. The increase in MRR is a substantial gain when considering Q & A systems, because MRR is serving as a proxy for the median position of a document that fulfills the users information need. However, $tf - idf$ weighting does not universally improve every metric, and $P@1$ stays the same when compared to the results generated by using raw term frequencies.

One problem with using standard $tf - idf$ weighting is that longer documents get higher $tf - idf$ scores by virtue of being longer. The thought behind this is that longer documents generally have more term occurrences than shorter documents, and this difference in term frequency will inflate the $tf - idf$ scores.

This can cause co-occurrences from longer documents to be weighted higher than co-occurrences from shorter documents, and this effect will bias the generated thesaurus towards longer documents.

4.2.3.2 Other Term Frequency Normalization Algorithms

Equation 4.5 is not the only way to adjust raw term frequencies, there are several popular methods that all curve raw term frequencies. Maximum term frequency normalization is likely the second most common way to curve raw term frequencies after the standard log method (equation 4.5). This method is outlined in equation 4.7.

$$ntf_{t,d} = \alpha + (1 - \alpha) * \frac{tf_{t,d}}{tf_{max}(D)} \tag{4.7}$$

Longer documents generally have higher term frequencies than shorter documents, so with any absolute scaling method (equation 4.5) longer documents will generate more significant co-occurrences than shorter documents by virtue of their larger term frequency values. Maximum term frequency normalization was created to address this problem, and the general intuition behind this method is to curve all term frequencies in a given document by the maximum term frequency present in the document. In theory this puts all documents on equal footing, where the most frequent term is assigned a value of 1.0 and all subsequent terms are scaled by the most frequent term no matter the length of the document.

Additionally, there is a smoothing term α in equation 4.7 that is usually set to a value between 0.0 and 1.0, but most papers use .4 which is the value this system used in its testing. The purpose of the smoothing term is to prevent small changes in tf from causing large changes in ntf.

Table 4.2 shows the effects of maximum term frequency normalization compared to the standard $tf - idf$ weighting. This weighting has a positive impact on $P@1$, MAP, and MRR, but has a marginal impact on *recall*. It also has a minimal runtime cost that occurs when constructing the term by document matrix. Overall, it has a positive impact on retrieval performance when used in an automatic thesaurus expansion methods.

4.2.3.3 Cosine Weight Normalization

There are a number of approaches that deal with $tf - idf$ weighting's inherent bias towards longer documents. As covered earlier, maximum term frequency normalization (subsubsection 4.2.3.2) is one possible approach to deal with this problem. Cosine normalization is another approach that operates on the full $tf - idf$ value instead of just the tf value. Instead of directly normalizing term frequencies, cosine normalization normalizes an entire document to unit length. This type of normalization guarantees that each document has a length of one which ensures that every document has the same potential to contribute to the final co-occurrences.

$$doc_{norm}(d_i) = \frac{\vec{d_i}}{|\vec{d_i}|} \tag{4.8}$$

A drawback of this type of normalization is that it does not distinguish between entries that have been inflated due to large term frequency values and entries that are significant due to their *idf* values. Hence, if two documents are the same length, but one contains rarer terms (terms with high *idf* values) the information is lost in this process. So not only does the system normalize documents of different lengths, it normalizes documents that have different term distributions.

Table 4.2 shows how cosine normalization affects the automatic thesaurus query expansion system. Compared to maximum term frequency normalization it performs worse overall, and, it shows small losses in $P@1$, MAP, and MRR over the plain $tf - idf$ system. This system likely performs worse than term frequency normalization due to its inability to distinguish between high $tf - idf$ scores resulting from large TF values or *idf* values. Even when maximum term frequency normalization is applied in conjunction with cosine normalization the results are still underwhelming. The best expansion results are achieved with maximum term frequency normalization on top of $tf - idf$ weighting.

4.2.4 Co-occurrence Cutoff and Effect on Expansion

The co-occurrence cutoff value is used to delineate between co-occurrences significant enough to be used as synonyms and co-occurrences that should be ignored. As the cutoff value decreases, more "synonyms"

are added to the co-occurrence matrix. Intuitively, as the number of synonyms increases *recall* performance should also increase, but inversely MAP will decrease as more noisy terms are added to the query. Generally, the cutoff value allows the user to choose balance between *recall* and MAP values, and as one metric increases the other usually decreases.

This is exactly the pattern that table 4.2 shows. Lower cutoff values show increased *recall*, but decreased $P@1$, MAP, and MRR. The sweet spot for the Medpedia corpus seems to be between .75 and .95, as values in this range show significant improvements to *recall* but do not sacrifice much in the way of MAP and MRR. Any values much lower than .75 start to degrade the precision metrics significantly. This type of degradation is expected as more noise is added the query in the form of automatically generated synonyms.

4.2.5 Using Document Co-occurrences to Construct Paragraph System Thesaurus

Generally, automatic thesauri are generated using the documents that are being indexed and retrieved. However in the case of the Medpedia corpus, the same content is available as both documents and paragraphs. As noted in the baseline results, document level *recall* is fairly good while paragraph level *recall* is relatively poor. It is possible to generate an automatic thesaurus using document level co-occurrence statistics and then use that thesaurus to expand queries in the paragraph system. This method is trying to leverage the good document retrieval performance to improve paragraph retrieval.

This thesaurus is generated in a manner identical to what was described in subsection 4.2.2. However, there is more variation among document lengths compared to the paragraph lengths, so the document co-occurrence matrix benefits significantly more from cosine normalization. Term frequency normalization also had worse performance on the document level thesaurus. Documents are significantly longer than paragraphs and there is a much larger discrepancy between the maximum occurring term and median occurring term in the document set when compared to the paragraph set. Under maximum term normalization, documents which have high term frequency outliers undervalue average terms since the normalization factor is so large. So this type of normalization marginalizes the values of terms that come from documents with skewed term distributions.

Table 4.3 shows the results of using document co-occurrences to generate a thesaurus for the paragraph retrieval system. Thesauri generated with document co-occurrences system are able to achieve higher P@1 and MRR values than the paragraph co-occurrence based systems. This improvement in precision is largely due to the incorporation of the additional semantic information gained by having co-occurrences from 60-70 related paragraphs grouped together in a semantically related collection (document). In the paragraph system, this information is thrown away, so the system has to generate a thesaurus based solely on the paragraph level co-occurrences. However, the paragraph co-occurrences system shows better *recall* at comparable MRR values when compared to the document co-occurrences system. This suggests that document co-occurrence based systems are not as useful as paragraph level co-occurrences when the result set is being reranked.

4.3 Paragraph System Results with Automatic Thesaurus Query Expansion

Considering the relatively small amount of information that this approach requires it performs well on the Medpedia corpus with the MiPACQ query set. Automatic document expansion and query expansion methods universally improved recall. Most of these methods should improve the MiPACQ reranker performance because they return more relevant documents than the baseline system without a significant drop in precision. Even though most systems did not cause significant drops in precision, many methods did cause slight drops in MRR, MAP, and P@1 while improving *recall*. Not every automatic expansion method performed identically, and the methods that were able to incorporate document level information showed higher maximum values for the precision metrics (P@1, MRR and MAP) than the methods that only used the paragraphs co-occurrences to perform expansion. The methods that used document level information (subsection 4.1.1 and subsection 4.2.5) were able to provide an eight to nine percent increase in recall with little or no drop in MRR. Additionally, one of these systems was able to provide solid MRR and P@1 gains over the baseeline system while still providing a seven percent increase in *recall* (subsection 4.2.5). All of these systems could help a Q & A pipeline produce better and more complete results.

The systems that just used paragraph level information also provided boosts to recall, and they generally had higher *recall* levels at comparable MRR values than the systems that incorporated document

level information. These systems would be most suited to Q & A pipelines, as these systems are able to provide higher *recall* at comparable MRR values than the systems based on document level information. The drawback of these systems is that the maximum improvement in $P@1$ and MRR is lower than when the document level systems are used. In general, document level systems are capable of providing a more "correctly" ranked set of documents, but a good reranker should show better results when working with the paragraph level systems than with the document level systems.

System	P@1	MAP	MRR	Overall Recall
Baseline Paragraph System	0.1379	0.0623	0.2265	0.6533
Paragraph System with Automatic Thesaurus Expansion with Raw Term Freqs (Cutoff 0.9)	0.1035	0.0376	0.1703	0.7167
Paragraph System with Automatic Thesaurus Expansion with tf-idf (Cutoff 0.9)	0.1035	0.0497	0.1956	0.7626
Paragraph System with Automatic Thesaurus Expansion with tf-idf and Term Frequency Normalization (Cutoff 0.9)	0.1379	0.0555	0.2156	0.7624
Paragraph System with Automatic Thesaurus Expansion with tf-idf Cosine Normalization (Cutoff 0.9)	0.1035	0.0482	0.1867	0.7608
Paragraph System with Automatic Thesaurus Expansion with Both (Cutoff 0.9)	0.0862	0.0466	0.1707	0.7519
Paragraph System with Automatic Thesaurus Expansion with tf-idf and Term Frequency Normalization (Cutoff 0.95)	0.1379	0.0581	0.2322	0.7604
Paragraph System with Automatic Thesaurus Expansion with tf-idf and Term Frequency Normalization (Cutoff 0.75)	0.1207	0.0536	0.1886	0.7848
Paragraph System with Automatic Thesaurus Expansion with tf-idf and Term Frequency Normalization (Cutoff 0.5)	0.1035	0.0493	0.1595	0.8515
Paragraph System with Automatic Thesaurus Expansion with tf-idf and Term Frequency Normalization (Cutoff 0.25)	0.0690	0.0308	0.0969	0.9236

Table 4.2: Automatic Thesaurus Generation for Query Expansion Results

System	P@1	MAP	MRR	Overall Recall
Baseline Paragraph System	0.1379	0.0623	0.2265	0.6533
Paragraph System with Automatic Thesaurus Expansion with tf-idf and Term Frequency Normalization (Cutoff 0.95)	0.1379	0.0581	0.2322	0.7604
Paragraph System with Automatic Thesaurus Expansion with tf-idf and Term Frequency Normalization (Cutoff 0.75)	0.1207	0.0536	0.1886	0.7848
Paragraph System with Automatic Thesaurus Expansion Doc Co-occurrences with tf-idf (Cutoff 0.9)	0.0862	0.0413	0.1527	0.7525
Paragraph System with Automatic Thesaurus Expansion Doc Co-occurrences with tf-idf and Term Normalization (Cutoff 0.9)	0.0690	0.0282	0.1133	0.8109
Paragraph System with Automatic Thesaurus Expansion Doc Co-occurrences with tf-idf and Cosine Normalization (Cutoff 0.9)	0.1379	0.0544	0.2269	0.7405
Paragraph System with Automatic Thesaurus Expansion Doc Co-occurrences with tf-idf and Both (Cutoff 0.9)	0.1207	0.0295	0.1458	0.7398
Paragraph System with Automatic Thesaurus Expansion Doc Co-occurrences with tf-idf and Cosine Normalization (Cutoff 0.95)	0.1552	0.0589	0.2542	0.7239
Paragraph System with Automatic Thesaurus Expansion Doc Co-occurrences with tf-idf and Cosine Normalization (Cutoff 0.85)	0.1207	0.0484	0.1884	0.7567
Paragraph System with Automatic Thesaurus Expansion Doc Co-occurrences with tf-idf and Cosine Normalization (Cutoff 0.8)	0.1207	0.0494	0.1901	0.8022
Paragraph System with Automatic Thesaurus Expansion Doc Co-occurrences with tf-idf and Cosine Normalization (Cutoff 0.75)	0.0862	0.0373	0.1482	0.8178

Table 4.3: Automatic Thesaurus Generation Using Document Co-occurrences for Query Expansion Results

Chapter 5

Resource Based Expansion Methodologies

Resource based expansion methods use an external resource such as an ontology to expand queries at runtime. Other possible resources include: thesauri, free text not included in the corpus, and formal semantic representations of information. For all of the resource based approaches in this paper SNOMED-CT (subsection 3.3.3) is used as the resource.

5.1 Naive Ontological Lexical Expansion System

This system is the baseline system for the subsequent improvements to the query expansion system. It does rudimentary query expansion using the LexEVS framework provided by the National Institute of Health. LexEVS is an open source ontology server that exposes a variety of APIs to load and access ontologies. This system is based on the SNOMED CT ontology distributed with the UMLS Metathesaurus. The query expansion is implemented as a new Lucene TokenFilter named SynonymFilter. This filter is combined with the rest of the standard filters (LowercaseFilter, StopwordFilter, StandardFilter, NumberFilter) to produce a SynonymAnalyzer. This analyzer is then used in the standard Lucene work flow described in the baseline paragraph system (section 3.7.2).

The SynonymFilter is implemented in Python and based on the LexEVS APIs. However, LexEVS is implemented as a Java application, so it is not possible to directly call into its APIs from Python. LexEVS does expose a REST interface from a JBOSS applet which is how the system accesses LexEVS from Python. The REST applet returns XML which is then processed via the standard ElementTree module in Python.

In order to do basic query term expansion, the SynonymFilter only examines one word at a time. This

means that the system will always do a poor job of expanding certain types of entities such as multi-word terms. For each word in the query, the system searches every entity definition in the SNOMED-CT ontology for potential matches. The system then takes every entity description match and retrieves its related entities. For each of these related entities the system retrieves a presentation list. In the UMLS Metathesaurus an entity's presentation list consists of the various ways that the entity usually shows up in text. The system then uses the contents of each entry in the presentation list to serve as synonyms for the query term. These synonyms are simply inserted in line after the query term to build the entire query string. Finally, the output of the SynonymFilter is fed through the StandardFilter, LowercaseFilter, and StopwordFilter to standardize the output synonyms.

Table 5.1 shows the performance of the naive ontological lexical expansion system. This system does not significantly improve *recall*, but does show small gains in MRR. *Recall* is not seriously improved because this system is not capable of generating a large number of synonyms due to its simple and limited process of mapping query terms to SNOMED-CT entities. The mapping process relies on query terms being present in entity definitions; however, entity definitions are often short so relying on query term and definition overlap is tenuous. Considering this limitation, this system is labeled as a "naive" approach when compared to the Metamap expansion method (section 5.2). In general this approach is quite limited in its ability to map query terms to SNOMED-CT entities, and overall expansion results suffer due to this problem.

5.2 Ontological Lexical Expansion using a NLP Pipeline (Metamap)

Metamap (subsection 3.3.1) is a NLP pipeline that maps free text to UMLS Metathesaurus entities. It is a good tool to address some of the shortcomings of the naive ontological expansion system. This Metamap expansion system uses the SNOMED-CT ontology in conjunction with the UMLS Metamap tool to identify SNOMED-CT entities in the MiPACQ queries, and this information is then used to expand queries.

To identify a concept, the entire free text query is sent to the Metamap tool. The Metamap tool parses the free text into phrases, and each phrase is tagged with a list of potential entity mappings. Each mapping is assigned a score between 1 and $1,000$ with higher scores representing a higher confidence in the correctness of the mapping. This confidence measure helps the expansion system decide whether the

mapping should be used for query expansion.

Once the relevant entities have been culled from the free text, they are used to generate additional terms for query expansion. The entities are turned into synonyms using the same process that is used in the "naive" expansion system (section 5.1). A list of related terms is retrieved for each entity, and the entity definitions and presentation lists from these terms are used to generate the final query expansion terms. These expansion terms are then appended to the original query terms, and the query is run through the system's standard Lucene work flow.

A couple of different methods for choosing which entity mappings to use for query expansion are explored in this paper, and the results from these methods are shown in table 5.1. The two broad approaches that were tested are described in the following subsections.

5.2.1 Metamap System Using the Top Mappings

One possible way to incorporate confidence information from Metamap is to take the top (first) N entities returned for a phrase. The entity list is sorted in descending order by confidence, so the first N entities returned by Metamap correspond to the N best matches for a phrase. This method has the advantage of always generating expansion terms whenever Metamap returns any entities. Additionally, when $N = 1$ this method guarantees that the entity picked is always the mapping that is most likely to be correct, and this helps minimize the noise added by incorrect mappings. However, when all of the returned phrase to entity mappings have low confidence ratings this system will still return at least one entity mapping. Often, this results in a decrease in retrieval performance for that query due to the noise introduced by using a poor mapping for expansion.

Table 5.1 shows the effect of the value of N on expansion performance. Selecting the single best entity mapping results in the best $P@1$, and MAP scores. Performance starts to decline as N equals three and more noisy entity mappings start to get introduced. At N equals five $P@1$ and MRR improve slightly over N equals three, but *recall* goes down as important terms start to become marginalized by all of the additional expansion terms. $N = 1$ seems to show the best retrieval performance for all resource based expansion methods. Overall, the resource based methods still perform worse than the best co-occurrence

based methods from chapter 4.

5.2.2 Metamap System Using a Threshold Value

Another possible way to incorporate confidence information from Metamap is to only include entity mappings above a certain threshold T. This has the advantage of guaranteeing that only good mappings will be used for expansion, but it also means that phrases with lots of ambiguity rarely get expanded due to their low confidence measures. Table 5.1 shows some results from using different threshold values. As the threshold value T decreases, noisier mappings are introduced which causes $P@1$, MAP, and MRR to decrease while *recall* goes up. When T equals 1000, the system performs similarly to selecting the best candidate, but it results in a slightly worse MAP value. In general, this thresholding method performs slightly worse than picking the top N entity mappings, and this discrepancy in retrieval performance is due to the thresholding system's tendency to miss ambiguous or low coverage terms.

System	P@1	MAP	MRR	Overall Recall
Baseline Paragraph System	0.1379	0.0623	0.2265	0.6533
Paragraph System with Naive Query Expansion	0.1379	0.0620	0.2350	0.6551
Paragraph System with Metamap Query Expansion (Top Candidate)	0.1035	0.0513	0.1675	0.7582
Paragraph System with Metamap Query Expansion plus Entity Presentations (Top Candidate)	0.1035	0.0484	0.1497	0.8052
Paragraph System with Metamap Query Expansion (Top 3 Candidates)	0.0862	0.0399	0.1468	0.7750
Paragraph System with Metamap Query Expansion (Top 5 Candidates)	0.1035	0.0398	0.1555	0.7640
Paragraph System with Metamap Query Expansion (Threshold of 1,000)	0.1035	0.0406	0.1679	0.7490
Paragraph System with Metamap Query Expansion (Threshold of 750)	0.0862	0.0373	0.1470	0.7532

Table 5.1: Resource Based Query Expansion Results

Chapter 6

Latent Semantic Indexing as an Alternative to Query Expansion

Latent semantic indexing is a different approach to how documents are indexed and searched. With the right corpus and right settings it can provide comparable results to a good query expansion method. The basic concept behind latent semantic indexing is to take advantage of "the implicit higher order structure in the association of terms with documents" [14]. This is accomplished by taking a standard term by document index and decomposing it into its left and right eigenvalues and its singular values. During this decomposition most of the eigenvectors corresponding to the smallest singular values are discarded, leaving a smaller dimensional space on the order of a few hundred dimensions. This smaller dimensional space is the latent semantic index (LSI), and the vectors in the space are semantic vectors. One of the purposes of collapsing the large term by document space into a smaller semantic space is to group terms that are semantically related together which helps with synonymy and polysemy. This type of indexing serves a similar purpose to query expansion, because synonyms should have similar representations in the latent semantic space. Hence, latent semantic indexing is another possible approach to dealing with query term and document term mismatch.

6.1 Creating the Latent Semantic Index

6.1.1 Creating the term by document matrix.

The first step in creating the latent semantic index is constructing the term by document matrix. Largely the construction is the same as described in subsection 4.2.1; however, instead of using pure term frequencies the term by document matrix is filled with $tf - idf$ values as described in subsection 4.2.3.1.

6.1.2 Decomposing the term by document matrix using SparseSVD.

Once the term by document matrix is constructed, it is fed to the sparseSVD python package. As described in subsection 3.4.3, sparseSVD is a python wrapper over SVDLIBC; a library for doing singular value decompositions on sparse matrices. The sparseSVD package takes the number of desired dimensions as an argument and returns two sparse matrices and a dense vector as a result.

$$U, S, V_T = \text{sparseSVD}(M, Dims) \tag{6.1}$$

U is a sparse matrix of term vectors in the latent semantic space, and V_T is the transpose of a sparse matrix of document vectors in the latent semantic space. S is a vector of length N of singular values sorted in descending order, where N is the number of dimensions in the latent semantic space.

The sparseSVD decomposition is very memory intensive which limits the size of matrices that can be decomposed and the dimensionality of the final result. On a test system with 32 GB of RAM it is able to perform the singular value decomposition of the term by document matrix when the requested number of dimensions is 3,000 or less.

6.1.3 Effect of the Dimensionality of the LSI on Retrieval Performance.

The number of dimensions of the latent semantic index has a large impact on retrieval performance. In the ideal case the number of dimensions matches the number of distinct semantic features in the corpus. However, it is rarely possible to know the number of distinct semantic features in a corpus a priori. Most of the seminal work on latent semantic indexing used dimension values ranging from 100 to 200 [14]. This work was mainly done on small document sets of 1,000-2,000 documents, but because the Medpedia corpus is 30 times larger than the corpora in the original paper, a larger dimensionality is likely required.

Table 6.1 shows the retrieval results for the LSI system. *Recall* performance roughly decreases as the dimensionality of the LSI increases. As the number of dimensions in the LSI approaches the rank of the original term by document matrix, *recall* performance should approach the baseline system performance. Empirically, retrieval performance improved for the LSI retrieval system as the dimensionality of the LSI

increased. Specifically, all three precision metrics ($P@1$, MAP, and MRR) improve as the LSI dimensionality increases. This suggests that the Medpedia corpus has at least 3,000 distinct semantic features, which is not surprising considering the corpus has over one hundred thousand unique terms. Unfortunately, the hardware used for testing the LSI system was not capable of supporting singular value decompositions with more than 3,000 dimensions, so the LSI system experiments never reached a point where the *recall* started to approach the baseline system's *recall*. Even with 3,000 dimensions, the LSI approach still falls significantly behind the other approaches covered in this paper. Because of its poor MAP and MRR values and its resource and time demands, latent semantic indexing is the least successful method for overcoming query term and document term mismatch.

6.2 Searching the Latent Semantic Index

In addition to modifying the term by document matrix, the search process also needs to be modified for latent semantic indexing. A query starts out in the standard term-document space, but it needs to be transformed into the semantic space before it can be compared to documents represented in the latent semantic index.

6.2.1 Projecting the Query into the Reduced Dimensionality Semantic Space.

To transform a query into the LSI semantic space, it is necessary to start with the SVD decomposition of the term by document matrix (equation 6.1). Once the decomposition has been computed it is possible to transform a query into this space. The math to incorporate a query into the LSI space was first described in [5].

$$M^T = (U * Diag(S) * V^T)^T \tag{6.2}$$

$$M^T = V * Diag(S) * U^T \tag{6.3}$$

$$M^T * U * Diag(S)^{-1} = V * Diag(S) * U^T * U * Diag(S)^T \tag{6.4}$$

$$V = M^T * U * Diag(S)^{-1} \tag{6.5}$$

V is the collection of k-dimensional document vectors in semantic space, hence to transform one document D the equation would look like equation 6.6.

$$D_k = D * U * Diag(S)^{-1} \tag{6.6}$$

D_k is one document out of the collection, and because queries can be treated like documents we get the final query transformation into semantic space.

$$Q_k = Q * U * Diag(S)^{-1} \tag{6.7}$$

Equation 6.7 shows the conversion from term space to semantic space for a query. Once the query is in semantic space, it can be compared to documents in semantic space using the standard cosine similarity measure.

6.2.2 Dense Cosine Comparison of the Query and Documents.

Once a query vector has been converted to the k-dimensional semantic space (equation 6.7) it is relatively straight forward to compare it to documents in the semantic space. Each document in the original term by document matrix ends up as a vector in semantic space. All of these vectors end up in V, so computing the relevance between a query and the document set ends up being a relatively easy calculation described in equation 6.8.

$$sim(\vec{Q_k}, \vec{D_k}) = \frac{\vec{Q_k} \cdot \vec{D_k}}{|\vec{Q_k}| * |\vec{D_k}|} \tag{6.8}$$

Equation 6.8 calculates the similarity between a query and document in semantic space, and the similarity value returned is used to rank the documents by relevance. Query and document pairs that have a negative cosine similarity measure in the semantic space are not included in the results.

6.2.3 Effect of Query and Document Weighting Schemes and Normalization Techniques on Retrieval Performance.

There are a variety of weighting schemes that can be applied to both documents and queries that affect retrieval performance. The default results in table 6.1 were run without query or document cosine weight normalization. Cosine weight normalization has little impact when applied to the free text query before it is transformed to semantic space, and it has a negative effect on retrieval performance when applied to the original term by document matrix. When applied to the term by document matrix, cosine normalization throws away some information necessary for making absolute comparisons between terms from different documents. This loss of information makes extracting good semantic features from the corpus more difficult, and it results in a poorer LSI overall. Hence, normalizing paragraphs of different lengths is not desired when constructing the latent semantic index. When cosine normalization is applied to a query it has no impact because it does not effect the relative term weights, and relative term weights is the only piece of information in a query that is used for document comparisons. Overall, cosine normalization does not seem to provide any benefits in the LSI domain.

Maximum term frequency normalization has a negative impact on LSI retrieval performance (section 4.2.3.2). Although this type of normalization is beneficial to eliminate retrieval bias towards long documents, it also makes less information available to create the latent semantic index. The system can no longer determine how many more times the top occurring term in $document_i$ occurs over the top occurring term in $document_j$, because both term occurrence values have been normalized to one. Without this information the generated LSI performs worse than an LSI based on a term by document matrix that uses standard $tf - idf$ values. Even though many of these normalization techniques had positive effects for automatic thesaurus generation systems, they do not perform as well for latent semantic indexing and retrieval. Largely, this is because the singular value decomposition needs very accurate document vectors in term space to produce an accurate decomposition, and because these normalization methods throw away some of the original document information they tend to produce poorer latent semantic indexes for the Medpedia corpus.

6.3 Paragraph System using LSI

Table 6.1 shows the retrieval results for the LSI system using different dimensions for the SVD decomposition. Generally, indexes created with fewer dimensions perform worse than indexes based on a higher dimensional space (subsection 6.1.3), and even high dimensional indexes perform worse than both the co-occurrence based expansion methods and resource based expansion methods described earlier in the paper. In theory, it should be possible to generate a latent semantic index with enough dimensions to provide good MAP, MRR, and $P@1$ performance without sacrificing too much of the *recall* gained by using a LSI system. However, our test system was not able to reach dimensions high enough to create a latent semantic index with reasonable MRR values. Additionally, if a latent semantic index were able to provide comparable results to the other methods, it would still be significantly slower and more resource intensive to run than any of the other query expansion approaches this paper has covered.

System	P@1	MAP	MRR	Overall Recall
Baseline Paragraph System	0.1379	0.0623	0.2265	0.6533
LSI Paragraph System with 50 Dimensions	0.0517	0.0100	0.0598	0.8537
LSI Paragraph System with 100 Dimensions	0.0173	0.0044	0.0372	0.7920
LSI Paragraph System with 150 Dimensions	0.0173	0.0033	0.0295	0.7653
LSI Paragraph System with 250 Dimensions	0.0345	0.0039	0.0383	0.7932
LSI Paragraph System with 500 Dimensions	0.0173	0.0064	0.0450	0.7758
LSI Paragraph System with 1,000 Dimensions	0.0517	0.0086	0.0748	0.7438
LSI Paragraph System with 2,000 Dimensions	0.0345	0.0116	0.0810	0.7459
LSI Paragraph System with 3,000 Dimensions	0.0517	0.0207	0.1009	0.7565
LSI Paragraph System with 1,000 Dimensions and Maximum Term Freq Normalization	0.0517	0.0083	0.0727	0.7329
LSI Paragraph System with 1,000 Dimensions and Document Cosine Normalization	0.0172	0.0085	0.0507	0.7352
LSI Paragraph System with 1,000 Dimensions and Query Cosine Normalization	0.0517	0.0086	0.0748	0.7438
LSI Paragraph System with 1,000 Dimensions and Doc and Query Cosine Normalization	0.0172	0.0085	0.0507	0.7352

Table 6.1: Latent Semantic Indexing Results

Chapter 7

Results Summary

All three approaches to query expansion examined in this paper improved *recall* performance over the baseline Lucene retrieval system. Most *recall* gains were in the 8% to 15% range, but some of the automatic thesaurus generation methods showed up to a 27% *recall* gain accompanied by a significant drop in MRR. The two methods that yielded the best MAP and MRR values used information about which parent document a paragraph was derived from to improve expansion performance. Both of these methods resulted in gains to either MAP or MRR over the baseline system and improved *recall*.

The co-occurrence based expansion methods performed best as an aggregate which is line with earlier work that suggested the co-occurrence statistics perform as well as or better than knowledge rich approaches. Both document expansion and query expansion systems were among the top performing expansion systems, and both systems produced similar results suggesting that document expansion can be a viable method for addressing synonymy and granularity issues in medical Q & A systems. Expansion systems that used SNOMED-CT and the UMLS Metathesaurus were able to provide significant gains to *recall* performance, but these gains came at the expense of MRR performance. However, future work exploring conceptual indexing using the UMLS Metathesaurus may improve these shortcomings [16].

Latent semantic indexing largely had poor retrieval performance compared to the other methods. *Recall* gains were similar to the knowledge poor and knowledge rich techniques, but MAP, MRR, and $P@1$ were all substantially lower using LSI than other methods at comparable *recall* levels. Additionally, LSI proved to be more computationally intensive than the other methods at both index and query time. Generally, LSI performance improved as dimensionality increased, but the computational cost of computing the SVD

of the term by document matrix also increased as dimensionality increased. This increased computation cost limited the size of the semantic space we could test, but given more computing resources LSI could potentially be competitive with the other expansion methods.

In systems that have access to document level information, generating an automatic thesaurus based on document co-occurrences to expand paragraphs provides the largest boost to MRR and MAP. However, since medical medical Q & A systems rerank the results outputted by these expansion methods, improving *recall* without losing too much precision was the goal of these systems. When considering this, generating an automatic thesaurus based on paragraph co-occurrences with maximum term frequency normalization should provide the most improvement to these Q & A systems, and for this corpus and query set it is the most effective system examined in this paper.

Chapter 8

Future Work

All of the approaches in this paper helped to address the query term and document term mismatch problem (synonymy and granularity). However, there are a few improvements to the query expansion systems tested in this paper that could yield further gains.

For the document expansion at index time methods, a more robust term selector could provide additional recall gains with little precision loss. Instead of selecting the parent terms that have the highest $tf - idf$ weights, an automatic summarization method could be used. This type of system would generate a few sentences from the parent document that would be indexed alongside the paragraph to help contextualize the paragraph. The additional and more robust context should provide performance improvements over the standard $tf - idf$ weight selection mechanism.

There are a few improvements to the automatic thesaurus generation systems that might yield additional gains as well. The current automatic thesaurus generation system expands every query term regardless of whether or not it is likely to need expansion. A potential improvement to this system would be to only expand query terms that have a low base rate of occurrence. This improvement should result in higher precision because it introduces less noise from expanded commonly occurring terms while maintaining the *recall* benefits of the standard system.

Another potential improvement to the automatic thesaurus generation system would be to tag terms with their word sense. This would allow the thesaurus system to distinguish between word senses and generate more accurate synonyms. However, this may not result in a large improvement in this domain due to medical terms primarily only having a single word sense. A related improvement would be to send

documents and queries through a named entity recognition system. This could allow the system to identify multi-word terms and expand only certain classes of words, and this type of selective expansion should help the automatic thesaurus systems to generate more consistent MRR and MAP gains.

Pseudo-relevance feedback is an approach that is not described in this paper, but it is designed to address the same type of query term and document term mismatch problems that the other approaches try to eliminate. It would be interesting to see how it performs against the methods tested in this paper. Additionally, pseudo-relevance feedback can be utilized after applying one of the query expansion techniques described in this paper. This could yield additional retrieval gains without much of a penalty to precision performance.

Due to the resource intensive nature of latent semantic indexing, the Medpedia corpus was only able to be decomposed via the singular value decomposition with 3,000 factors or less. However, as the number of factors in the SVD increases, the retrieval performance of the latent semantic index also increases. It would be enlightening to see how the latent semantic index behaves as the number of dimensions returned by the SVD increases.

Another retrieval model based on semantic concepts labeled by the UMLS Metamap program could have a positive impact on retrieval performance. In this type of model the documents and queries are sent through the Metamap program to identify the relevant UMLS Metathesaurus semantic concepts. These concepts would then serve as the basis of a new retrieval system that retrieved and ranked documents based on their semantic concept overlap [16]. This type of indexing might be a more effective use of the information contained in the UMLS Metathesaurus ontology.

The Metamap resource based query expansion system was based on a subset of the entire UMLS Metathesaurus for performance and efficiency reasons. However, using the entire Metathesaurus for expansion could yield better retrieval results due to the expanded term and presentation coverage. It should be relatively easy to conduct this experiment because it does not require any changes to the Metamap system other than loading LexEVS with the full UMLS Thesaurus.

Chapter 9

Conclusion

A variety of query expansion methods were evaluated on the MiPACQ query set using the Medpedia document set in this paper. The main goal of this evaluation was to find a suitable query expansion method for medical Q & A systems, specifically the MiPACQ system. MiPACQ employs a reranker on top of a natural language processing pipeline and standard information retrieval system. The performance of the reranker is dependent on the documents returned by the underlying standard information retrieval system. This retrieval system is based on the vector space retrieval model, and it had low *recall* performance using the MiPACQ queries on the Medpedia paragraph corpus. Largely the low *recall* performance is caused by query term and paragraph term mismatch, which can be addressed by query expansion methods.

In this context, three query expansion varieties were tested and evaluated: knowledge rich query expansion approaches that use an external knowledge source to expand queries; global co-occurrence based approaches that use term co-occurrences in the corpus to expand queries; and latent semantic indexing which transforms the documents and queries into a semantic space. Each approach was evaluated against four metrics, *recall*, MAP, MRR, and P@1. Emphasis was placed on the *recall* results as the paragraphs returned by the retrieval system will be reranked by the MIPACQ reranker.

Query and document expansion systems that include parent document information when expanding paragraphs yielded the best performance on precision metrics. While the co-occurrence based expansion system that only used paragraph co-occurrence statistics had the best overall retrieval performance. Both of these systems add minimum runtime cost to query processing and searching, and both systems have parameters that allow the user to tweak the *recall* and *precision* balance easily.

Overall, query expansion can provide significant retrieval performance gains, and is suitable for medical Q & A applications. The most successful methods increase *recall* and MRR with no penalties to the MAP or $P@1$ metrics. Future work should examine how these systems perform as part of medical Q & A system pipeline.

Bibliography

[1] Eneko Agirre, Xabier Arregi, and Arantxa Otegi. Document expansion based on wordnet for robust ir. In Proceedings of the 23rd International Conference on Computational Linguistics: Posters, COLING '10, pages 9–17, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[2] Alan R. Aronson. Effective mapping of biomedical text to the umls metathesaurus: The metamap program. 2001.

[3] Alan R Aronson and Francois-Michel Lang. An overview of metamap: historical perspective and recent advances. 2010.

[4] Alan R. Aronson and Thomas C. Rindflesch. Query expansion using the umls. 1997.

[5] M. W. Berry, S.T. Dumais, G.W. O'Brien, Michael W. Berry, Susan T. Dumais, and Gavin. Using linear algebra for intelligent information retrieval. SIAM Review, 37:573–595, 1995.

[6] J. Bhogal, A. Macfarlane, and P. Smith. A review of ontology based query expansion. Information Processing and Management, 43(4):866 – 886, 2007.

[7] Matthew W. Bilotti, Boris Katz, and Jimmy Lin. What works better for question answering: Stemming or morphological query expansion? In Proceedings of the Information Retrieval for Question Answering (IR4QA) Workshop at SIGIR 2004, 2004.

[8] Justin Zobel Bodo Billerbeck. Document expansion versus query expansion for ad-hoc retrieval. Proceedings of the Tenth Australasian Document Computing Symposium, pages 34–41, 2005.

[9] Brian L. Cairns, Rodney D. Nielsen, James J. Masanz, James H. Martin, Martha S. Palmer, Wayne H. Ward, and Guergana K. Savova. The mipacq clinical question answering system, September 2011.

[10] C. G. Chute and Y. Yang. An evaluation of concept based latent semantic indexing for clinical information retrieval. Proc Annu Symp Comput Appl Med Care, page 639643, 1992.

[11] Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. Latent semantic kernels. Journal of Intelligent Information Systems, 18:127–152, 2002. 10.1023/A:1013625426931.

[12] Carolyn J. Crouch and Bokyung Yang. Experiments in automatic statistical thesaurus construction. In Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '92, pages 77–88, New York, NY, USA, 1992. ACM.

[13] C.J. Crouch. An approach to the automatic construction of global thesauri. Information Processing and Management, 26(5):629 – 640, 1990.

[14] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE, 41(6):391–407, 1990.

[15] William Hersh, Susan Price, and Larry Donohoe. Assessing thesaurus-based query expansion using the umls metathesaurus. In In Proc. of the 2000 American Medical Informatics Association (AMIA) Symposium, pages 344–348, 2000.

[16] Diem Thi Hoang Le, Jean-Pierre Chevallet, and Dong Thi Bich Thuy. Thesaurus-based query and document expansion in conceptual indexing with umls: Application in medical information retrieval. In RIVF'07, pages 242–246, 2007.

[17] Zhenyu Liu and Wesley W. Chu. Knowledge-based query expansion to support scenario-specific retrieval of medical free text. In Proceedings of the 2005 ACM symposium on Applied computing, SAC '05, pages 1076–1083, New York, NY, USA, 2005. ACM.

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

[19] Xiangming Mu and Kun Lu. Towards effective genomic information retrieval: The impact of query complexity and expansion strategies. J. Inf. Sci., 36(2):194–208, April 2010.

[20] Roberto Navigli, Paola Velardi, Dipartimento Informatica, and Roma La. An analysis of ontology-based query expansion strategies. Information Retrieval, pages 42–49, 2002.

[21] US National Library of Medicine. Umls - about. Electronically, March 2012. http://www.nlm.nih.gov/research/umls/about_umls.html.

[22] Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, SIGIR '03, pages 41–47, New York, NY, USA, 2003. ACM.