Information Retrieval for Clinical Decision Support

Adam Rhine

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science

University of Washington

2017

Committee:

Meliha Yetisgen

Fei Xia

Program Authorized to Offer Degree:

Linguistics

University of Washington

Abstract

Information Retrieval for Clinical Decision Support

Adam Rhine

Chair of the Supervisory Committee:
Meliha Yetisgen, PhD
Biomedical and Health Informatics

Within the field of clinical support, access to relevant, peer-reviewed information from medical journals and other research publications is critical to making informed decisions regarding the diagnosis and care of patients. This study aims to build a complete biomedical information retrieval system, based on the data released by the National Institute of Standards and Technology for Text REtrieval Conference Clinical Decision Support track. Through the use of query expansion, machine-learning classification, a vector space model with tf-idf ranking implementation, along with additional specialized preprocessing data transformation and post-processing scoring techniques, we constructed an IR system competitive with the state of the art for this specific retrieval task, and demonstrated our success through the use of standardized evaluation metrics.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Leaping into the TREC challenge was an intimidating prospect, particularly in a field dominated by much larger and more experienced institutions. Only through immense research efforts, stringent time management, and the assistance of the UW advisors and faculty was it possible to successfully construct an original, powerful, and competitively viable clinical information retrieval system.

This study would not exist without Chris LaTerza. His Java prowess got our codebase off the ground, and his research was the driving force behind our unique system architecture. I would also like to express tremendous gratitude to Meliha Yetisgen, whose expertise and insight helped transform a simple class project into a globally competitive clinical IR system. Thanks to Fei Xia, for her critical analysis, hard-hitting questions, and valuable thesis content suggestions, and to Brandon Graves, for his technical assistance with the linguistics server. Finally, thank you to my family, and to Jessica and Jinx Chan, for supporting me through the long hours and sleepless nights.

Chapter 1. INTRODUCTION

Clinical decision making is generally guided by three distinct factors: patient/clinician preferences (based on cultural beliefs, personal values, education, etc.), outside constraints (e.g. formal policies/laws, community standards, and financial pressures), and evidence (e.g. patient data, clinical research, or systematic reviews) [1]. This third category is of particular interest in the field of clinical informatics, which focuses on the problems of extracting and analyzing clinical data to better assist in patient care.

The data utilized by clinical informaticians can be subdivided into two categories: patient-specific information and knowledge-based information [1]. Each data type presents its own unique challenges. Patient-specific information (i.e. patient medical records) is often difficult to access, and typically is based on unstructured clinical notes whose cadence and style vary wildly depending on the clinicians or clinical system involved. On the other hand, knowledge-based information - collected from medical journals and other research publications - is more structured, but the deep breadth and depth of available resources can make it difficult to find data relating to a patient's specific health context.

Health informatics-related information retrieval attempts to bridge the gap between these two disparate clinical data categories. Information retrieval (IR) refers to the task of retrieving a specific and relevant data set from a larger collection of resources. If the ultimate goal of searching for health information is to make a decision, such as whether or not to order a test or prescribe a treatment, then IR systems in the health informatics field aim to utilize patient-specific data to find the most applicable knowledge-based information for answering these clinical decision questions.

Our research into in this field focused on examinations of information retrieval methods, along with specialized IR implementations for bioinformatics and clinical decision. In Chapter 2, we discuss this literature in depth, including examinations of the use of these IR approaches by existing TREC teams (as discussed in their own published works). Chapter 3 explains our own system methodologies, and Chapter 4 reviews the results of this system as recorded through slightly varying trials. Chapter 5 discusses these results in further detail, with direct comparisons to the results of other TREC teams. Finally, Chapter 6 concludes the study, and includes potential areas of future work.

1.1 TREC Clinical Decision Support Track

TREC, or the Text REtrieval Conference, is a workshop series sponsored by the National Institute of Standards and Technology (NIST) centered on large-scale evaluation of information retrieval methodologies [9]. The TREC program includes a number of different research branches, including Federated Web Search, Contextual Suggestion Support, and Temporal Summarization. Our focus in this study is on the Clinical Decision Support (CDS) track, an annual workshop with the goal of evaluating biomedical literature retrieval systems for providing answers to clinical decisions regarding patient cases. When provided with a list of clinical notes and question types ("What is the patient's diagnosis", "What diagnostic tests are appropriate for this patient", or "What treatment is appropriate for this patient"), for each note/type combination, CDS participants are challenged with retrieving the most relevant biomedical articles from a document set of approximately one million texts [12].

While our system build is generic and modular enough to work with any given year's CDS task, our variables and weights were tuned to the 2015 document set and topic list B, which at

the time of writing is the most recent CDS task with a publically available gold standard evaluation document list [9].

1.2 2015 TREC PARTICIPANTS

In preparation for building our system, we examined the released research papers from a number of participants in the 2015 challenge. The systems submitted from the Democritus University of Thrace (DUTH) [4], Wayne State University (WSU-IR) [5], and the Athens University of Economics and Business (AUEB) [6] were of particular interest, not only because their high evaluation scores - especially WSU-IR, whose system outperformed almost all others in the 2015 task - but also because of the variability of their approaches. The DUTH team, for example, built separate document indexes for each question type, while the AUEB team integrated bigrams into their index and explored some retrieval methods with weight model variants, and the WSU-IR team implemented specialized query weights and a Markov Random Fields-based retrieval model. More detail surrounding these teams' approaches, and their relation to our system, can be found in the chapters that follow.

Chapter 2. LITERATURE REVIEW

From an abstractive perspective, we broke the TREC task into three basic categories: (1) the query formulation, (2) the document index, and (3) the actual search and score algorithm. In the literature review below, relevant research papers, web articles, and other publications are examined and discussed, with particular regard for their application to our potential IR system. This review specifically focuses on the categories of retrieval algorithms (which incorporate indexing and scoring methods) and common query formulation approaches, with additional discussion of Lucene, a Java library that allows for various indexing and scoring implementations.

2.1 RETRIEVAL ALGORITHMS

2.1.1 *Boolean Retrieval*

Boolean retrieval is the most basic form of information retrieval, yet it remains surprisingly commonplace. In fact, Manning et al point out that that the majority of law librarians still recommend the use of boolean retrieval to search legal texts, with the results simply sorted by publication date [2].

At its core, boolean retrieval involves indexing each document in the database with a boolean value for each potential query term, in what is commonly referred to as a binary term-document incidence matrix.

Table 2.1. Incidence Matrix of the works of Shakespeare against a limited term list

	Antony and Cleopatra	**Julius Caesar**	**The Tempest**	**Hamlet**	**Othello**	**Macbeth**
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

One advantage of this approach is that you can easily incorporate boolean operators, such as AND, OR, and NOT, into your search query. In Figure 2.1 above, a search on the terms ‘Brutus AND Caeser AND NOT Calpurnia’ could be converted into a binary equation (with ‘NOT Calpurnia’ represented by the inverse of ‘Calpurnia’) of ‘110100 && 110111 && 10111’. The result of this operation would be 100100, returning *Antony and Cleopatra* and *Hamlet* as the retrieved texts. While there is not any ranking or nuance to the result list of a boolean search, the accuracy of your results will be very high (assuming a very strict acceptance criteria) [2].

An even more efficient method of boolean retrieval, as pointed out by Manning et al, is to create an inverted index, which assigns numeric identifiers (IDs) to each document, and stores lists of document identifiers (docIDs) as values in a dictionary, where the keys are the potential search terms.

Table 2.2. Inverted index of the works of Shakespeare against a limited term list.

Dictionary		**Lists**
Antony	→	1, 2, 6
Brutus	→	1, 2, 4
Caesar	→	1, 2, 4, 5, 6
Calpurnia	→	2
Cleopatra	→	1
mercy	→	1, 2, 4, 5, 6
worser	→	1, 2, 4, 5

With an inverted index, the previously presented search would simply find which docIDs are present in the Brutus and Caeser lists, but not in the Cleopatra list, and compile the cross-referenced doc lists into a query result [2].

While a binary search against an inverted index is efficient, it is not practical for the purposes of the TREC retrieval task. To start, it is missing the essential scoring component required to rank the document search results, a requirement of the task. The boolean retrieval approach also has no functionality for compartmentalizing and assigning different weights to different fields in the search query and/or document set. Furthermore, on a more general level, boolean retrieval suffers from a severe lack of complexity - for instance, the use of AND operators produces high precision but low recall searches, while the OR operators produce high recall but low precision searches, and it is nearly impossible to find a satisfactory middle ground [2]. It is no surprise, then, that we found no past TREC teams which opted for a boolean query approach. However, it does provide a meaningful introduction into the information retrieval

concept, and the inverted index (extended to incorporate the frequencies of each term) can still be implemented with more complex search models.

2.1.2 *Vector Space Model*

Unlike boolean retrieval, the vector space model approach does allow for ranked retrieval and a more nuanced result list. In a vector space model comparison each document is represented as a vector, in which each dimension/feature corresponds to a term in the vocabulary. These features are each assigned a weight for that term, (which can be calculated using a number of approaches, such as binary, raw term frequency, *tf-idf*, etc), and features that are not present are treated as having a weight of 0. The search query is also represented as a vector, as if it were a short document. These vectors - the document and the query - are compared using a document similarity equation (such as cosine similarity), which provides a numeric score. Document scores are then ordered and compiled into a ranked list of the relevant results [2].

The vector space model of document/query comparison, specifically implemented with the cosine similarity equation, was the most commonplace search approach among the examined TREC 2015 research papers [4][5][6]. The Athens University of Economics and Business team even makes specific note of how, between all their difference index/search attempts, the vector space model consistently proved the most effective [6].

2.1.3 *Weighted Zone Scoring*

Within the scoring equation, as Manning expounds on, further variance can be added through the use of weighted zone scoring. Zones are defined as divisions of the free text, usually through document metadata - for example, the title of a document might be stored in one zone, and the author in another. Each of these zones is assigned a relative weight g in the range [0,1], where

the sum of all these zone weights is equal to 1. Given a scoring function *s*, a weighted zone score for a query-document pair can then be defined as [2]:

$$\sum_{i=1}^{l} g_i s_i$$

Equation 2.1. Weighted zone scoring equation

2.1.4 *Tf-idf*

Tf-idf, or "*term frequency-inverse document frequency*", is one of the more common weighting schemes used by the vector space model. Term weights are calculated by finding the number of times a given term appears in a specific document, and multiplying it by the inverse of the number of documents containing the term divided by the total number of documents in the doc set (typically scaled logarithmically) [2]. A typical *tf-idf* weighting equation, as outlined by Manning, resembles the following:

$$\text{tf-idf} = \text{tf}_{d,t} \times \log \frac{N}{\text{df}_t}$$

Equation 2.2. Standard *tf-idf* equation

Manning provides further examples of *tf-idf* variants, as illustrated in table 2.3 below, which contains potential replacements for sections of the standard equation.

Table 2.3. Term frequency, document frequency, and normalization substitutions/variations

Term frequency		Document frequency		Normalization	
n (natural)	$\text{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\text{tf}_{t,d})$	t (idf)	$\log \frac{N}{\text{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \text{tf}_{t,d}}{\max_t(\text{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\text{df}_t}{\text{df}_t}\}$	u (pivoted unique)	$1/u$ (Section 6.4.4)
b (boolean)	$\begin{cases} 1 & \text{if } \text{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}, \alpha < 1$
L (log ave)	$\frac{1+\log(\text{tf}_{t,d})}{1+\log(\text{ave}_{t \in d}(\text{tf}_{t,d}))}$				

The Manning text notes that *tf*, *idf*, and normalization variants maintain relative strengths and weaknesses, depending on the given IR task. For instance, in an average document query task utilizing the standard *tf-idf* equation, if a document *d* returns a score of *x*, a document of *d'*, consisting of *d* added to itself, would return a score of 2*x*. The use of the maximum tf normalization formula in eq. 2.3 removes this anomalous scoring.

$$\mathrm{ntf}_{t,d} = a + (1 - a)\frac{\mathrm{tf}_{t,d}}{\mathrm{tf}_{\max}(d)'}$$

Equation 2.3. Maximum *tf* normalization formula

However, it also introduces a number of alternate issues, like allowing term weightings to be dramatically altered by only a slight change in the stop word list [2]. As such, these equation variants are not necessarily always beneficial, and any approach using them would likely require ample testing against the baseline *tf-idf* equation.

One example of an applied *tf-idf* variant can be found in the AUEB team's attempts at the TREC 2015 task. The team compared two different approaches: a standard vector space model using *tf-idf* weighting and cosine similarity, and a similar model that instead used a *tw-idf* weighting model, where *tw* is a score measuring the importance of each term in the graph-of-words representation of documents [6]. However, while the AUEB team's *tw-idf* model was certainly interesting and unique, it ultimately performed significantly worse than their *tf-idf* approach, thus demonstrating the efficacy of the standard formula.

2.2 Query Formulation

The extraction and/or expansion of queries seemed to be the least-explored, or at least least-effectively utilized, aspect of the TREC 2015 task, based on the DUTH, WSU, and AUEB research papers [4][5][6]. All three teams turned to the Unified Medical Language System

(UMLS), a knowledge base of biomedical concepts collected by the National Library of Medicine [10]. UMLS is constructed from a combination of different biomedical vocabularies, and its sources are updated quarterly. Metamap is a commonly used NLP tool that maps biomedical text to biomedical concepts in the UMLS metathesaurus [11].

As far as the utilization of the UMLS vocabularies by the TREC teams, the AUEB team simply noted that they utilized UMLS "for automatic query expansion," with no actual details on the particulars of this implementation [6]. On the other hand, DUTH dives into detail on their UMLS query expansion methods, which involved filtering out any non-medical words from the summary, utilizing UMLS Metamap to extract concepts from these filtered search strings, and then retrieving synonymous terminology for those concepts, which was finally added back into the original summaries. However, the DUTH expanded queries (which also included standardized gender and age group extractions) performed significantly worse than the unmodified queries [4]. Similarly, the WSU team implemented a query expansion method using both unigrams and bigrams of the UMLS concepts, with said concepts given various term weights within the search algorithm. But as with the DUTH team, WSU's UMLS-based expanded queries ultimately scored lower than other approaches [5]. Given the expansive nature of the document set, it stands to reason that incorporating synonyms and concepts would improve the scores, but based on past teams' attempts this area clearly requires further research and testing.

2.3 LUCENE

A common approach amongst the 2015 TREC submissions was the use of Lucene [8], an information retrieval Java library, as the preferred method of both indexing and querying. Released as an open source project in 2000, Lucene has grown immensely in popularity over the

last decade, and is now considered the most widely used IR library in the world [3]. It provides a simple and easy jumping off point - only a handful of classes are needed to integrate Lucene into existing systems - yet it also contains a number of more complex extensions and modules that provide deep options for customization and specialization. Lucene also makes no assumptions about the data used in indexing and searching, allowing for a broad range of applications [3].

2.3.1 *Scoring Function*

Both the DUTH team [4] and the AUEB team [6] mention their implementation of Lucene within their systems; the AUEB team even makes particular note of how they used Lucene's default scoring method as part of their *tf-idf* search approach [6]. The equation used in Lucene's default scoring method can be found in equation 2.4, where *coord*() refers to a built-in scoring function that returns higher values based on how many query terms the document contains, *queryNorm*() is a score normalization factor, *getBoost*() is a user-set query term boost value set at search time, and *norm*() is a function that includes the user-set field boost value set at index time [3].

$$\text{score}(q,d) = \text{cood}(q,d)\cdot\text{queryNorm}(q,d)\cdot \sum_{t \text{ in } q} (\text{tf}(t \text{ in } d)\cdot\text{idf}(t)^2\cdot t.\text{getBoost}()\cdot\text{norm}(t,d))$$

Equation 2.4. The standard Lucene VSM scoring function

The proven efficacy of the default Lucene scoring method from AUEB's trial results [6] suggest it to be an ideal starting point for our approach, possibly tested against slight variations from the *tf-idf* variant chart in table 2.3.

2.3.2 *Field Indexing*

Included in the Lucene Java library implementation is the functionality to create not only an inverted index, but one that also allows for multi-field indexing within each document. The

greatest advantage of multi-field indexing is the ability to filter out sections of a document at index time, so that only the document's relevant information is queried against. Additionally, by splitting a document into separately indexed fields, said fields can be given individual weights, thus granting them greater or lesser effect within the actual search queries (as shown in the Lucene scoring equation in eq. 2.4) [3].

Selecting only the pertinent document sections, and tuning the field weights of those sections within the index, can greatly improve the relevancy of the query results. The DUTH team demonstrated the strengths of multi-field indexing in their TREC 2015 approach. Instead of indexing the entire document set, the DUTH system extracted only the Title, Category, Abstract, Keywords, and Body sections of the documents, and indexed those with (unspecified) unique weights [4]. Surprisingly, the DUTH query against their multi-field index actually performed poorly in comparison to a simple, full-doc index [4]; however, this is possibly due to additional transformations that were added to the DUTH multi-field index, or poor weighting values, and does not necessarily discount the entire approach.

Chapter 3. METHODOLOGY

The overall goal of our study was to build an information retrieval system for medical articles based on the TREC 2015 task data. We approached this task by combining our knowledge of Natural Language Processing (NLP) with input gleaned from the TREC 2015 result papers and other related readings.

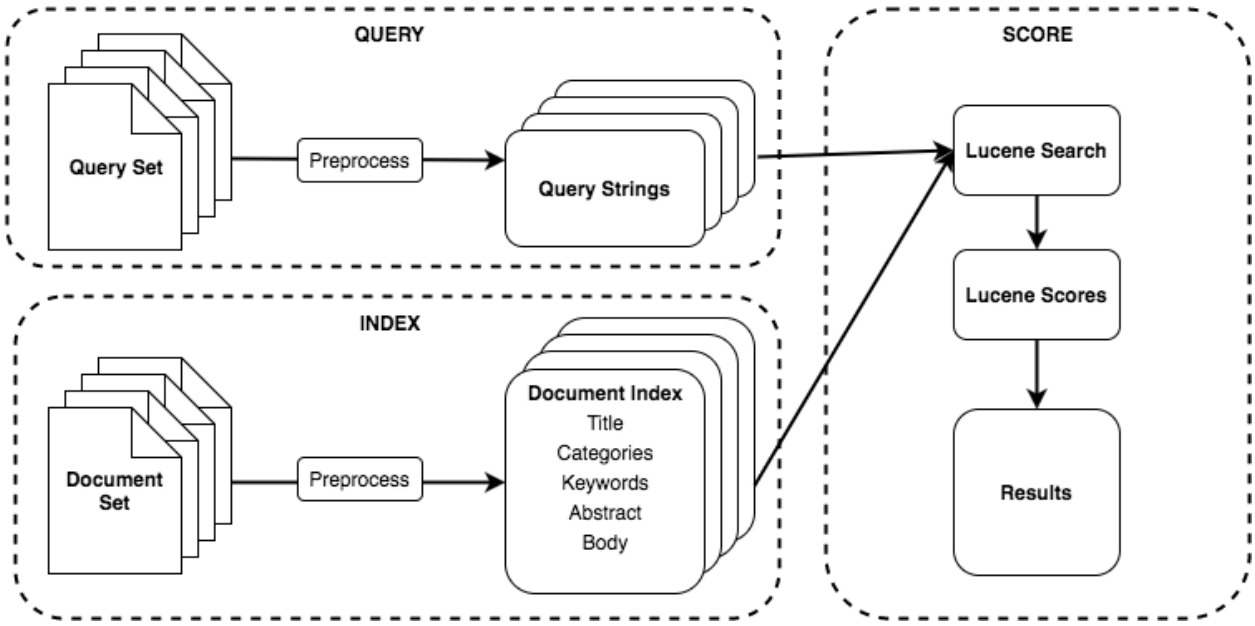

Figure 3.1. Basic system overview

Our initial baseline extraction system is presented in Figure 3.1. The overall architecture of the system described in this document is presented in Figure 3.2. In the following sections, each major component will be explained in detail.

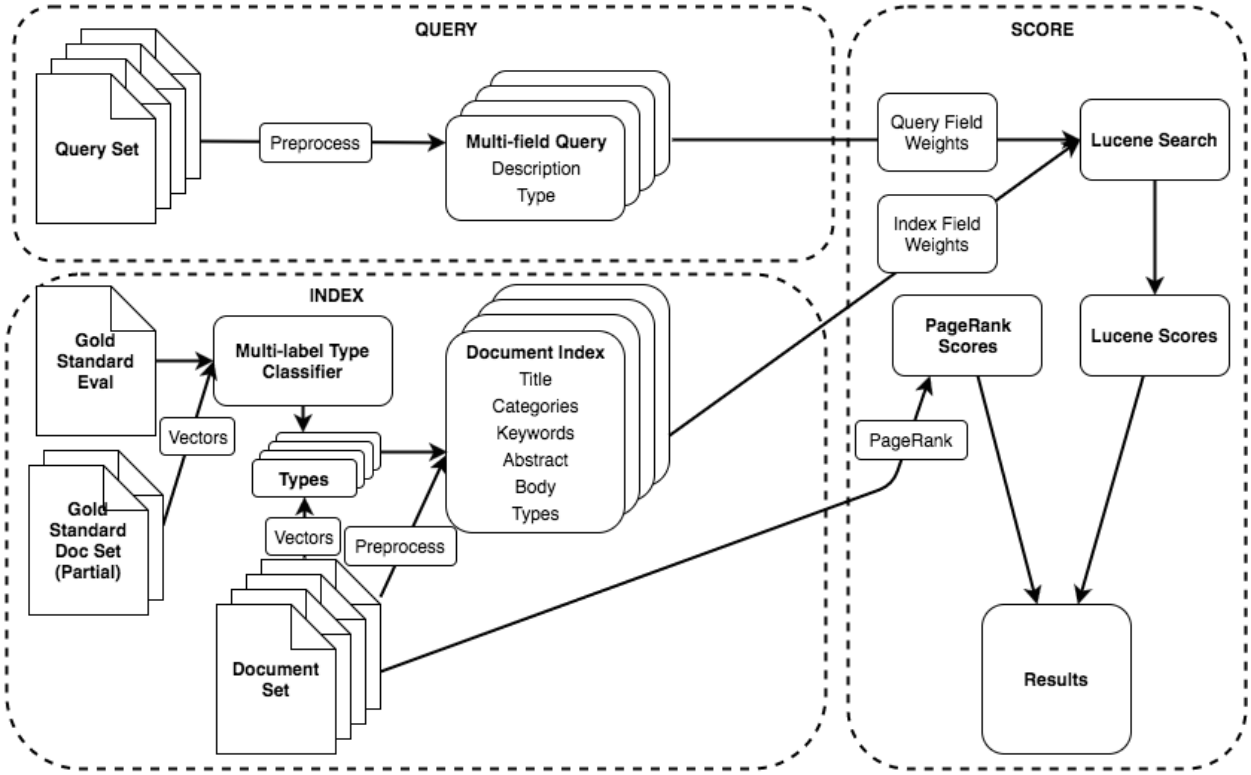

Figure 3.2. Current system overview

3.1 Document Index

One of the most common themes in our readings was the construction of an inverted index for more efficient searching. However, rather than simply dumping our 1 million count document set directly into an index, we first implemented a number of preprocessing steps on our documents, so that the information indexed (a) consisted of just the most relevant data for our retrieval task, (b) contained markers tailored to different query types, and (c) was indexed in a manner that lends itself to more complex scoring techniques, such as field-specific search weights.

3.1.1 *Field Parsing*

Parsing our index into multiple fields provides a number of advantages over a straight document index. First, it assures that only information relevant to clinical decision support is incorporated into our index.

Figure 3.3. Document XML data extraction and preprocessing

There are a number of XML fields in the initial documents - like author names, copyright information, and other article metadata - that contain irrelevant or near-irrelevant information for our information retrieval task, which we filter out during our preprocessing and field parsing step.

Field parsing also allows us to map particular fields in the query to particular fields in the index. In other words, when calculating the *tf-idf* score of a specific term in the query, instead of using the entire term set of the documents to derive *tf* and *idf*, you can limit the calculation to the term set of a specific indexed field within those documents. We utilized this functionality in our

approach by strictly matching the Type field of our query to the Type field of our index (see 3.2.2 for more information).

Finally, because we are utilizing the standard Lucene vector space model scoring function, parsing our index data into multiple fields allows us to assign variable scoring weights to those fields. Fields like the Abstract - which contains a much higher ratio of relevant to non-relevant unigrams than the Body - can be assigned a higher weight, leading to a better final document selection. Our semi-optimized weight distribution, found using trials of our base system with Type classifier, can be found in table 3.1 below.

Table 3.1. Index field weights

Index Field	Title	Abstract	Keywords	Categories	Body	Type
Weight	3.5	1.5	3.0	2.0	1.0	2.0

When Lucene calculates the total document scores (see eq. 2.4), each term's individual *tf-idf* score incorporates the term's index field weight in the *norm(t,d)* function.

3.1.2 *Multi-label Type Classifier*

Most of the fields within our document index were extracted directly from the document XML. However, examination of past TREC data and approaches suggested a significant difference in the documents expected for the three different query Types (Diagnosis, Test, and Treatment). In order to better tailor our results to the query type, we implemented a MaxEnt Type classifier, trained against a set of documents labeled using the 2014 gold standard evaluation results. From these result lists, 30,121 unique documents were found, a randomly selected 80% of which were used as training data.

As we needed to account for documents belonging to multiple types, the classifier utilizes a One vs. All approach to labeling. In other words, labels are derived using three separate binary

classifiers, each of which labels a document as either the given Type (e.g. Diagnosis), or 'blank' Type. The three binary classifiers' results are ultimately compiled into one Types field, which is added to that document's index – thus, as a result, a given document could be labeled with anywhere from zero to all three of the provided Types.

To compile our training data, for each document found in the 2014 gold standard evaluation document list, we created a feature vector from the article's Title, Abstract, Keywords, Categories, and Body fields. The feature vectors pulled were simple binary unigram lists, i.e. the unique single tokenized terms from the documents' fields (with punctuation removed).

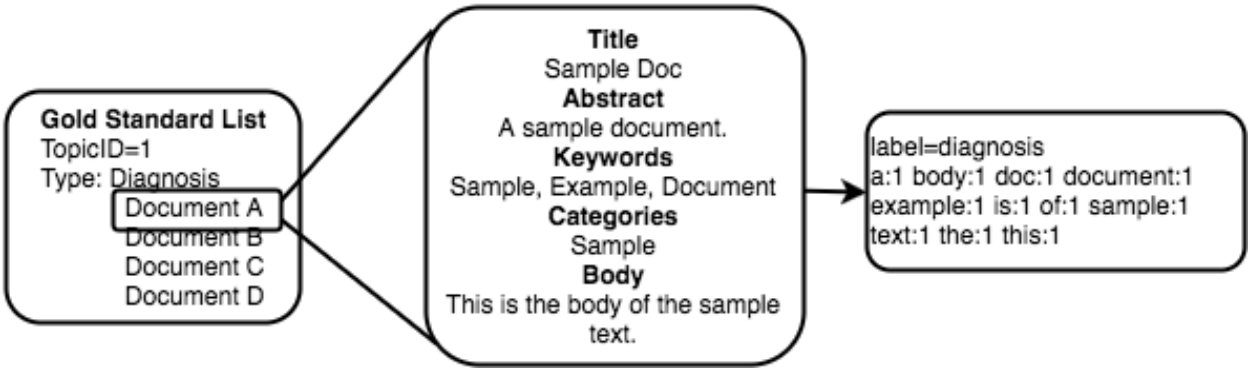

Figure 3.4. Feature vector extraction

For each Type, feature vectors were created for the entire training document set, where documents that appeared in that Type's topic list were labeled with the Type name, and all other documents were labeled with a negative label. From the gold standard evaluation list, the exact breakdown of labeled documents can be found in Table 3.2 below.

Table 3.2. Gold standard document type label counts

Diagnosis Only	Test Only	Treatment Only	Diagnosis & Test	Diagnosis & Treatment	Test & Treatment	All Types
8957	9172	7989	1230	1232	991	550

The binary MaxEnt classifiers were then created using these Type features, and their efficacy was tested with the remaining 20% of the gold standard document list. The accuracy metrics of these test documents are shown in Table 3.3.

Table 3.3. Classifier test data accuracy metrics

Diagnosis	Precision	0.80557
	Recall	0.84337
	F-measure	0.82404
Test	Precision	0.87862
	Recall	0.88908
	F-measure	0.88382
Treatment	Precision	0.88098
	Recall	0.87516
	F-measure	0.87806

Specific binary type label breakdowns (positive vs. negative) are calculated in Table 3.4 below.

Table 3.4. Positive/Negative type label breakdown

Diagnosis	Positive	11969
	Negative	18152
Test	Positive	11943
	Negative	18178
Treatment	Positive	10762
	Negative	19359

The topics upon which the gold standard lists were based are unrelated to their individual Type. In other words, there is nothing unique to the clinical notes of each specific Type, other than the specific Type label (see fig. 3.5).

```
<topic number="8" type="diagnosis">
<summary>
A 10-year-old boy with difficulty concentrating, daytime sleepiness,
and failure to thrive. The boy sleeps restlessly, snores, sweats,
breathes heavily through his mouth and gasps in his sleep.
</summary>
</topic>
<topic number="9" type="diagnosis">
<summary>
A 10 year old child with recent history of pork consumption presents
with fever, myalgia, facial edema and eosinophilia
</summary>
</topic>
<topic number="10" type="diagnosis">
<summary>
A 38 year old woman with severe dysmenorrhea, menorrhagia, and
menometrorrhagia. PMH of infertility treatment and ectopic pregnancy
</summary>
</topic>
<topic number="11" type="test">
<summary>
A 56-year old Caucasian female presents with sensitivity to cold,
fatigue, and constipation. Physical examination reveals hyporeflexia
with delayed relaxation of knee and ankle reflexes, and very dry
skin.
</summary>
</topic>
<topic number="12" type="test">
<summary>
A 44-year-old man complains of severe headache and fever. Nuchal
rigidity was found on physical examination.
</summary>
</topic>
<topic number="13" type="test">
<summary>
A 5-year-old boy presents with difficulty in breathing, stridor,
drooling, fever, dysphagia and voice change.
</summary>
</topic>
```

Figure 3.5. Cross-type topic summary comparison

Because of this uniqueness in the document Type categories, along with the fact that the documents selected were spread over ten different unrelated sets of clinical notes, the highest rated features for each classifier were generally terms and concepts specific to that Type. For

example, some high-impact features for the Test-Type classifier included procedures and experiments (e.g. “angiogram”, “colposcopy”,” echocardiogram”) along with physical characteristics to test (e.g. “reflexes”, “rigidity”, “responsiveness”).

3.2 Query Formulation

The queries (or “topics”) provided in the TREC Clinical Decision Support task consist of the following XML fields/metadata:

- **Topic Number**
- **Topic Type**: Diagnosis, Test, or Treatment
- **Description**: Admission notes “translated” into more common and/or layman-friendly terminology
- **Summary**: 1-2 sentence summary of admission notes
- **Disease (2015 task only)**: The doctor’s diagnosis. Only present in Test and Treatment topic types.

Generally, per the TREC instructions, any or all of the above data can be used in the retrieval task. The only exception is with the Summary, Description, and Notes (in the 2016 task) fields; of these three admission notes fields, only one can be used within each system run. However, multiple system runs can be submitted, with each run potentially using a different admission note field. Outside of that limitation, we had free reign to incorporate any techniques or data into the creation of our IR queries.

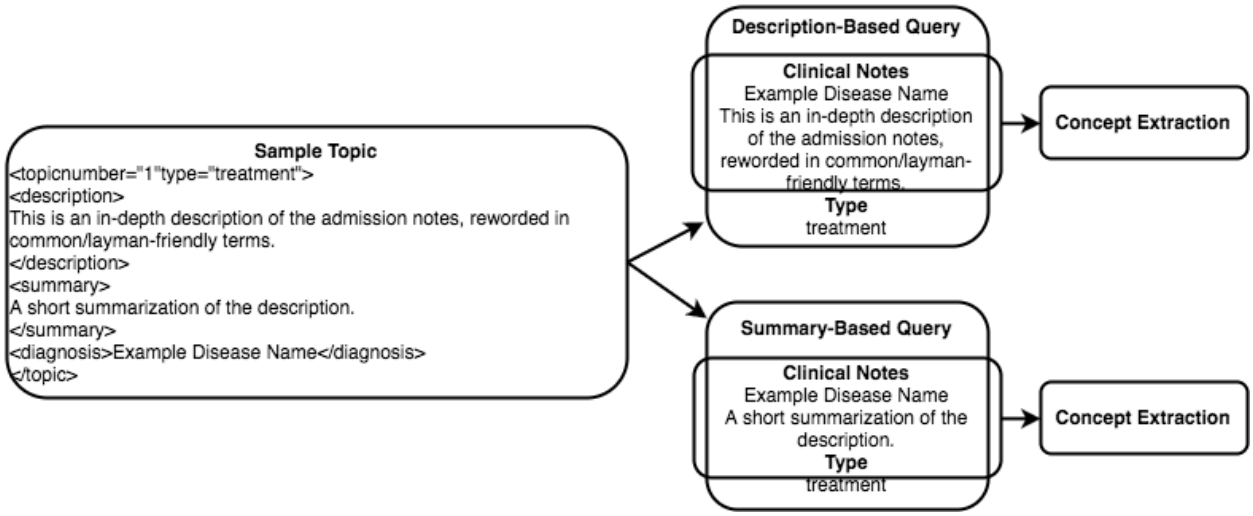

Figure 3.6. Query formulations from topic XML

3.2.1 *Concept Extraction*

We experimented with a number of concept extraction approaches as part of our query formulation. One of the more basic techniques was the normalization of gender and age. The query descriptions and summaries are based on modified doctor's notes, and as such contain non-uniform language patterns to describe the same features. In particular, the vocabulary for gender shows a wide variability, with words like "man," "male," "boy," and "male child," for example, used to describe similarly gendered patients. We attempted a number of approaches to make these terms more generic across the queries, including the incorporation of all terms into each query, transforming the gender references into one standard set of terms, or removing the gender references altogether.

In the same vein, the queries include age references for the patients that are very specific ("32-year-old", "55 yo", etc). We hypothesized that these specific ages were generally unlikely to appear in most of the relevant research documents. For normalizing the age information, we tested both systematically replacing the ages with age group markers ("elderly", "adult", "teen", etc), as well as simply removing the age completely.

Finally, based on our readings from past TREC team systems, one common thread was the use of the Unified Medical Language System (UMLS), a metathesaurus of biomedical terminology. The advantage of incorporating UMLS into our query formation is that our query string will have a higher chance of finding relevant documents if it includes the health terminology synonyms and/or concepts. On the other hand, including too broad a selection of synonyms can cause an excess of noise, resulting in lower scores for relevant documents and higher scores for potentially irrelevant documents. As such, the key to successfully incorporating UMLS into our query set is finding the correct balance of filters on the concepts/synonyms - a balance that we are still trying to find.

At the time of this writing, every attempt (or combination of attempts) at concept extraction has resulted in a significantly lower score for our system. This is in contradiction with our extensive literature review, which suggests that concept extraction, particularly via UMLS, can be applied beneficially to the clinical decision support information retrieval task. To that end, we plan to experiment with more advanced and complex concept extraction techniques as we continue to build upon our system (see Conclusion section for more details).

3.2.2 *Field Parsing and Mapping*

Similar to the field parsing used in the index (see section 3.1.1), we looked to incorporate field parsing into our query creation as well. Most importantly, as with the index, parsing allows us to assign varied weights to different parts of the query, allowing us to raise or lower the relevance of certain query sections depending on their importance. The semi-optimized weight distribution of our query fields can be found in Table 3.5 below.

Table 3.5. Query field weights

Query Field	Clinical Note	Type
Weight	1.0	4.0

These query term weights are then incorporated into the Lucene scoring function (eq. 2.4) as *t.getBoost()*.

Individual field parsing also allows us to map each field in the query to specific field(s) within the document index, as shown in figure 3.7.

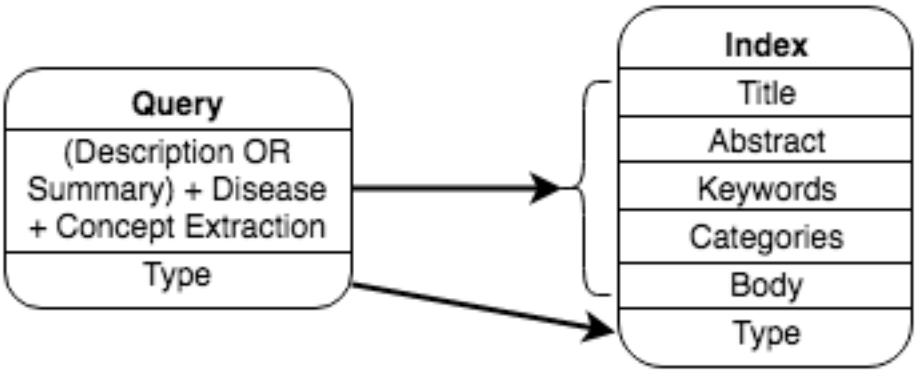

Figure 3.7. Query-Index field mappings

Our queries consist of two separate fields. The clinical note string is formulated by taking either the provided clinical Description or Summary, adding the disease name if provided, and then applying any concept extraction techniques (age/gender normalization, UMLS concept expansion, or none). When the individual scores of each term in this clinical notes field are calculated, the *tf* and *idf* formulas use the term sets from all the fields in the indexed documents except Type. In contrast, the second field in our queries was the Type field, whose string was extracted directly from the provided topic, and whose *tf-idf* value is calculated by only looking at the terms in the Type field of the indexed documents.

Our initial tests, which simply added the Type string to the clinical notes field, then used that single query string to search against all the indexed fields without any Type labeling of the

document index or mapping of the index-query fields, actually lowered our scores. It was only by adding the Type field to the document index, and then mapping the Type field of the query so that the search only queries with that field in the Type field of the index, that our scores were drastically improved (see Results section for more details).

3.3 Search and Scoring

3.3.1 *Lucene Scoring*

Once our query strings and full index are established, our system uses the standard Lucene vector space model scoring function to retrieve, score, and rank the relevant documents from our index. The exact scoring equation can be seen in figure 2.4. The query and index field weights are determined through our own testing, although we plan to incorporate a systematic approach to find the ideal weights in the future (see Conclusions).

The TREC evaluation document lists are not of standard length, but rather seem to just incorporate the manually determined "most relevant" documents, resulting in document lists ranging from 500-1500 documents per topic. To that end, rather than simply returning the X most relevant documents from our Lucene query, we take each topic's highest-scoring document, and return all documents whose score is at least X% of that highest-scoring doc. These lists are compiled and outputted in the expected TREC evaluation format, lists of document IDs (and their respective scores and ranks) split by topic.

3.3.2 *PageRank*

The final step of our result compilation is the incorporation of PageRank scoring into our final document result rankings. PageRank, first developed as part of the Google search algorithm, is

an algorithm that creates a probabilistic distribution to represent the likelihood of ending up on a given page by clicking random links [7].

In our system, rather than linking web pages through hyperlinks, we connect documents from our initial document set via the References section of the documents. In other words, our PageRank output represented the probability of ending on a given research paper by following random references from other papers.

Our PageRank construction is only possible due to the structured XML nature of the provided documents. We were able to easily extract the references from each document, create a cross-reference table of the document titles/years and their document ID, and then utilize the standard PageRank algorithm to determine each document's PageRank score.

Once the final PageRank scores are calculated, our system scales these scores by a semi-optimized scaling factor (which was estimated at 10^6 through multiple rounds of testing – see Results section for more details) and adds them to the previous Lucene results scores.

$$\text{Score}_d = \text{Luc}_d + (\text{Scale} \cdot \text{PR}_d)$$

Equation 3.1. Final scoring equation

The document list is then re-ranked and reorganized based on the new total scores to determine our final result lists.

Chapter 4. PERFORMANCE EVALUATION

4.1 EXPERIMENTS

In order to gather thorough data that reflects the efficacy of the different modules in our system, we ran 10 trials - 5 using the Description fields of the provided TREC topics, and 5 using the Summary fields. Each 'Base' trial run was a run of the base system, as described in figure 3.1. Further trials implemented some combination of Query Expansion (Age/Gender Normalization), Query Expansion (UMLS), the MaxEnt Type Classifier, and the PageRank scoring algorithm, all of which are shown in the system overview in figure 3.2.

Table 4.1. Experiment trials

	QE (A/G Norm.)	**QE (UMLS)**	**MaxEnt Type Class.**	**PageRank**
DescBase				
DescM			✓	
DescMP			✓	✓
DescMPQE1	✓		✓	✓
DescMPQE2		✓	✓	✓
SumBase				
SumM			✓	
SumMP			✓	✓
SumMPQE1	✓		✓	✓
SumMPQE2		✓	✓	✓

4.2 EVALUATION

In the evaluation, final document lists for each topic are compared against gold standard evaluation lists, which are established by the Department of Medical Informatics of the Oregon Health and Science University (OHSU) [12]. There are four key evaluation metrics obtained through this doc list comparison:

- **Inferred Average Precision (infAP)**: Approximates the average precision even when the relevance judgments are incomplete.
- **Inferred Normalized Discounted Cumulative Gain (infNDCG)**: Approximates the usefulness, or gain, of a document based on its position in the result list using the graded relevance scale of the assessors.
- **R-Precision (R-prec)**: The precision after R documents have been retrieved, where R is the number of relevant documents for the topic.
- **Precision at Rank 10 (P@10)**: The fraction of articles within the top-10 results which the doctor judges as relevant documents.

4.3 RESULTS

Our highest-performing trials were those using the Summary-based queries, rather than the Description-based queries. For the infAP, infNDCG, and R-Prec metrics, the addition of the MaxEnt Classifier and the PageRank scoring led to the highest evaluation scores across all trial runs. In terms of the P@10 metric, our best performance was achieved by the base system, with no additional features. The lowest evaluation scores for infNDCG and R-Prec were from the base

Description-based trial, and the lowest infAP and P@10 evaluation scores were in the Description-based trials with query expansion.

Table 4.2. Experiment trial results

	infAP	**infNDCG**	**R-prec**	**P@10**
DescBase	0.0272	0.2215	0.2494	0.1400
DescM	0.0345	0.2592	0.2918	0.1333
DescMP	0.0345	0.2592	0.2917	0.1333
DescMPQE1	0.0339	0.2580	0.2893	0.1267
DescMPQE2	0.0267	0.2291	0.2551	0.1300
SumBase	0.0352	0.2578	0.2867	**0.1933**
SumM	0.0425	0.2903	0.3275	0.1900
SumMP	**0.0426**	**0.2904**	**0.3276**	0.1900
SumMPQE1	0.0402	0.2782	0.3125	0.1567
SumMPQE2	0.0305	0.2425	0.2659	0.1767

Chapter 5. DISCUSSION

In the TREC 2015 published participant results, teams were ranked by their highest infNDCG evaluation score from their entire trial set [12]. When listed alongside these published scores, our system fares relatively well.

Table 5.1. TREC 2015 participant results (subset of all 2015 participants)

Participant System	**infNDCG**
Top Score	0.3821
WSU-IR	0.3246
Overall TREC 2015 Median Score	0.3095
Our System	**0.2904**
AUEB	0.2898
Overall TREC 2015 Mean Score	0.2870
DUTH	0.2394

While not the top, our system outperformed the overall mean infNDCG score for the 2015 teams, as well as the DUTH and AUEB teams.

Focusing on our system's individual trial scores, we can draw some measurable conclusions as to the efficacy of certain system features and enhancements. For example, unexpanded, smaller queries (those that used the shorter Summaries instead of the longer Descriptions, and did not include any query expansion techniques) scored significantly higher than their longer counterparts. Incorporating additional related terms, whether provided by the clinical note Descriptions or through concept extraction and expansion, aimed to draw in further relevant

documents for each task, but it instead appears to have actually diluted our queried document pool with unwanted results.

Meanwhile, introducing the MaxEnt multi-label type classifier noticeably improved our infAP, infNDCG, and R-prec scores, but actually lowered our P@10 scores. Our P@10 scores were relatively poor to begin with, suggesting that while our result doc set contains a high percentage of relevant documents, our relative ranking approach (particularly for the top listed documents) needs further enhancement. Furthermore, it appears that adding Type as a factor in the query/indexing provides an even worse document ranking for the top of the result document set. Yet the tremendous gains over the other three evaluation metrics demonstrate the value of using this type classifier for our IR task.

The PageRank algorithm, on the other hand, has only a slight effect on the overall evaluation scores. This is due in part to the *scaling variable* value of the PageRank scores. Through our testing, we found that the level of scaling applied to the PageRank scores before adding them to the Lucene scores could heavily modify the final evaluation results, but in most cases this influence was negative.

Table 5.2. PageRank scaling value results

Scaling Value	**infAP**	**infNDCG**	**R-prec**	**P@10**
0	0.0425	0.2903	0.3275	0.1900
$5x10^4$	0.0425	0.2903	0.3275	0.1900
$1x10^5$	0.0425	0.2903	0.3275	0.1900
$5x10^5$	0.0426	0.2903	0.3276	0.1900
$1x10^6$	0.0426	0.2904	0.3276	0.1900
$5x10^6$	0.0427	0.2904	0.3279	0.1833
$1x10^7$	0.0426	0.2901	0.3284	0.1833
$5x10^7$	0.0408	0.2863	0.3263	0.1467
$1x10^8$	0.0371	0.2811	0.3237	0.1267

Table 5.2 above demonstrates the effect of various scaling weights upon the final metric results. In the ideal weight range (~1,000,000), infAP, infNDCG, and R-prec are all slightly positively influenced, with no noticeable change to P@10 scores. Slightly higher in the scaling range (5,000,000-10,000,000), we see further slight improvements to the first three metric scores, but P@10 begins to suffer. Above that scale range, all scores begin to rapidly decline.

In general, our system appears well built in terms of both indexing and searching/scoring. The multi-field split and individualized weights of the document section provides a strong indexing foundation, and the incorporation of the derived Type field (as established by the trained MaxEnt classifier) noticeably improves upon that baseline. Similarly, the Lucene implementation of the *tf-idf* based vector space scoring equation outputs strongly scored result lists, which are further boosted by the query field search weighting (particularly for the Type

field), as well as the addition of the scaled PageRank document scores. However, our query formation is still lacking, and all trials attempting to incorporate new query-enhancing features resulted in lower evaluation metric scores across the board.

Chapter 6. CONCLUSION AND FUTURE WORK

Our evaluation results were very competitive for the 2015 task, and our relative trial results suggest that we have made large strides in our indexing and searching systems. However, our approach is constantly evolving, and there remain numerous features and improvements we still aim to implement within our CDS IR system.

One particular challenge that arose throughout our development process was balancing the ideal variable combinations. Each step in the process involves its own unique variables - the query and index fields each have distinct weights, individual query terms can have their own weights, the PageRank scores are multiplied by a relative scaling value before getting added to the Lucene scores, etc. As modifying any given variable can affect all subsequent step outputs, these parameters (numbering about 20 in total, excluding individual query term weights) exist in an intertwined state, and even small modifications of these parameter values can impact the final evaluation scores tremendously.

Unfortunately, when including the document set preprocessing and indexing step, a full system run can take hours to complete, so exhaustive parameter optimization approaches like grid searching [13] were impractical in our case. In the current system state, the parameter values were found using a limited, semi-random search [13] over a 15-20 trials, but in the future - particularly with more time and a fully-automated (instead of manually triggered) process, we plan to implement an optimization framework around the entire system in order to find the highest scoring parameter set possible.

An additional area of improvement we are still exploring is query expansion. As mentioned in section 3.2.1, and evidenced in our evaluation results, our current query concept extraction

techniques (the addition of UMLS concepts via Metamap, and the normalization of age and gender) are flawed to the point of noticeably decreasing our system scores. Yet our literature review of published TREC 2015 team papers, along with our personal research into UMLS, suggest that a proper query expansion approach can greatly improve both the precision and recall of the search result set. To that end, we are currently exploring new approaches in this area of our system, including filter-based UMLS term inclusion/exclusion and the incorporation of additional medical terminology databases, such as the BioMed Central Cases database [14].

BIBLIOGRAPHY

[1] William Hersh. 2009. Information Retrieval: A Health and Biomedical Perspective. Springer-Verlag New York, New York.

[2] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2009. An Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK.

[3] Michael McCandless, Erick Hatcher and Otis Gospodnetic. 2010. Lucene in Action, Second Edition. Manning Publications Co., Greenwich, CT.

[4] George Drosatos, Stefanos Roumeliotis, Avi Arampatzis, and Eleni Kaldoudi. 2015. "DUTH at TREC 2015 Clinical Decision Support Track". Democritus University of Thrace, Xanthi, Greece. http://trec.nist.gov/pubs/trec24/papers/DUTH-CL.pdf

[5] Saeid Balaneshin-kordan, Alexander Kotov, and Railan Xisto. 2015. "WSU-IR at TREC 2015 Clinical Decision Support Track: Joint Weighting of Explicit and Latent Medical Query Concepts from Diverse Sources". Wayne State University, Detroit, MI. http://trec.nist.gov/pubs/trec24/papers/wsu_ir-CL.pdf

[6] Giannis Nikolentzos, Polykarpos Meladianos, Nektarios Liakis, and Michalis Vazirgiannis. 2015. "AUEB at TREC 2015: Clinical Decision Support Track". Athens University of Economics and Business, Athens, Greece. http://trec.nist.gov/pubs/trec24/papers/DBNET_AUEB-CL.pdf

[7] Sergey Brin and Lawrence Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". Computer Science Department, Stanford University, Stanford, CA. http://infolab.stanford.edu/~backrub/google.html

[8] The Apache Software Foundation. 2011-2016."Apache Lucene." https://lucene.apache.org/

[9] National Institute of Standards and Technology. 2000. "Text REtrieval Conference: Overview". http://trec.nist.gov/overview.html

[10] Olivier Bodenreider. 2003. "The Unified Medical Language System (UMLS): Integrating Biomedical Terminology". National Library of Medicine, Bethesda, MD. http://nar.oxfordjournals.org/content/32/suppl_1/D267.short

[11] Alon R. Aronson. 2001. "Effective Mapping of Biomedical Text to the UMLS Metathesaurus: the MetaMap Program". National Library of Medicine, Bethesda, MD. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2243666/

[12] Kirk Roberts, Matthew S. Simpson, Ellen Voorhees, and William R. Hersh. 2015. "Overview of the TREC 2015 Clinical Decision Support Track". National Institute of Health, Bethesda, MI. http://trec.nist.gov/pubs/trec24/papers/Overview-CL.pdf

[13] James Bergstra and Yoshua Bengio. 2012. "Random Search for Hyper-Parameter Optimization". Department of Information, University of Montreal, QC, Canada. http://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

[14] Yue Wang and Hui Fang. 2014. “Exploring the Query Expansion Methods for Concept Based Representation”. Department of Electrical and Computer Engineering, University of Delaware, DE. http://trec.nist.gov/pubs/trec23/papers/pro-udel_fang_clinical.pdf