

ENTITY FINDER: A SYSTEM FOR ENTITY WEB PAGE RETRIEVAL USING
PSEUDO-RELEVANCE FEEDBACK

BY

RUI WANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Professor Chengxiang Zhai

ABSTRACT

Collecting all the online information about a particular entity (e.g., a person or a product) is a task commonly needed in many applications. In many cases, there often already exists a database with limited information about interesting entities, but those databases usually suffer from incompleteness and out-of-date problems. But with the increasing amount of information available on the World Wide Web, crawling and searching the web may be an attractive technological approach that can help update a database to make it more complete and up to date.

In this thesis, we propose a retrieval system that crawls and searches the web in order to complete and update the information about an entity already existent in a database maintained by an organization. Taking the information stored in the database as input, this system can crawl the web and retrieve the web pages mentioning the entities in the database. We study several approaches to solving this special retrieval problem, and propose a novel pseudo-relevance feedback approach to improve the retrieval accuracy. We evaluate our system over a dataset containing 112 alumni in the College of Engineering of the University of Illinois, and show that our system can effectively retrieve relevant pages of alumni on the web and that the novel pseudo-relevance feedback method outperforms a simple baseline approach.

ACKNOWLEDGEMENTS

I would like to express greatly appreciation to my advisor, Prof. Chengxiang Zhai. He has been incredibly helpful not only in my thesis work but also in doing computer science research in general. He guided me through this thesis from the very beginning and provided me useful advice especially when I was lost. Moreover, he shared his research experience and lessons he learned to me and other members in our research group. It has been an honor to work with him.

Also, thank my friend Fan Fu for support on writing an effective web crawler that greatly helps on data collection.

Finally, thank computer science department of the University of Illinois Urbana Champaign for providing great resources and giving me excellent experience in both undergraduate and graduate study.

TABLE OF CONTENTS

| | |
|---|----|
| CHAPTER 1 INTRODUCTION..... | 1 |
| 1.1 Background | 1 |
| 1.2 Related Works..... | 2 |
| 1.3 Contributions | 4 |
| CHAPTER 2 PROBLEM OVERVIEW | 7 |
| CHAPTER 3 SYSTEM DESIGN AND IMPLEMENTATION..... | 9 |
| 3.1 Web Crawling..... | 9 |
| 3.2 Indexing and Searching | 10 |
| 3.3 Result Integration | 14 |
| CHAPTER 4 PSEUDO-RELEVANCE FEEDBACK | 16 |
| CHAPTER 5 EVALUATION | 19 |
| 5.1 Dataset..... | 19 |
| 5.2 Experiments and Results..... | 20 |
| CHAPTER 6 CONCLUSION AND FUTURE WORK | 24 |
| REFERENCES | 26 |

CHAPTER 1

INTRODUCTION

1.1 Background

Many organizations or enterprises are maintaining information systems or databases that store information about some entities (e.g., a person or a product). Since those entities are often having significant meanings the task of maintaining information about those entities has been important to those organizations or enterprises. However, there are many challenges and problems in achieving this task. Firstly, the information stored in the databases or information systems they currently have is often limited. Incompleteness has always been a problem they are facing. Out-of-date issue is another crucial challenge because after the initial entering of the data, limited effort could be made to update the information about the entities in their databases.

One example of the task we mentioned above is to keep track of the alumni information by universities after their students graduate. There are many good reasons of doing this such as staying connected with alumni in order to let them to give back donation. The universities often have databases storing basic information about alumni entered at the time they graduate. But since students only provide limited information and they don't often update it is hard for the universities to gather complete and the most up-to-date information of their alumni.

With the rapid growth of the amount of information available on the web, it is reasonable and possible to explore technological approaches that collect information from the web to

solve the challenges described above. We may treat the web as a big source of information and our goal is to find the relevant data that could be integrated into the database we currently have. In the example of maintaining alumni database, it is reasonable to believe that more information could be discovered from the web such as social network profiles and web news articles that could be used to enrich our knowledge about the alumni. Therefore, crawling and searching the web pages that relevant to the entities in order to gather more complete and up-to-date information may be an attractive approach.

In order to find relevant web pages about the entities, building an information retrieval system that automates this process is needed. This consists different steps including web crawling, indexing and searching. In the process of searching, a query is needed served as input, and the system returns a ranked list of web pages. Instead of getting the query from human, our system needs to systematically construct the query itself based on the information we currently have in our database. However, many important terms may be missing in the query the system constructs due to the shortage of information. To solve this problem, query expansion [1] is used. In particular, a method called pseudo-relevance feedback (PRF) [1] has been the most effective. The system we propose utilizes this method in a novel way to improve the retrieval accuracy. In related works, more detailed discussion is provided.

1.2 Related Works

Web information retrieval is a well-studied area and many previous works have been done. Our system mainly explores pseudo-relevance feedback used to do query expansion in order to

improve the retrieval accuracy. This method is a revised version of relevance feedback [2], which lets users (i.e., human) manually annotate the retrieved documents as relevant or non-relevant. Then the retrieval system computes a better query representation based on user annotation and retrieves a different set of documents. While in pseudo-relevance feedback [2], the retrieval system automates this annotation process by assuming that the top K retrieved documents are relevant so that human effort is no longer needed.

Pseudo-relevance feedback has been implemented in many different retrieval models. Our system adopts it in vector space model [3] with a novel improvement by leveraging the database information we currently have. The baseline approach is based on Rocchio Algorithm [4].

In vector space model, documents and queries are represented as vectors. Each dimension is corresponding to a term. Each value (i.e., term weight) is computed by TF-IDF scheme [2]. The model ranks documents based on the similarity between the query vector and the document vector. More detail on implementation of the retrieval model is discussed in Chapter 2.

Rocchio Algorithm implements relevant feedback based on this model. The key idea is to form a new query vector by maximizing its similarity to relevant documents and minimizing its similarity to non-relevant documents. The formula that computes new query vector $\overrightarrow{Q_{new}}$, given the set of relevant documents D_r and the set of non-relevant documents D_{nr} is as follows [2]:

$$\overrightarrow{Q_{new}} = (a \cdot \overrightarrow{Q_o}) + (b \cdot \frac{1}{|D_r|} \cdot \sum_{\overrightarrow{D_j} \in D_r} \overrightarrow{D_j}) - (c \cdot \frac{1}{|D_{nr}|} \cdot \sum_{\overrightarrow{D_k} \in D_{nr}} \overrightarrow{D_k})$$

Where $\vec{Q_o}$ denotes the original query vector and a, b and c are weights for original query, relevant document set and non-relevant document set respectively. In other words, the new query vector is computed by moving the original query vector away from the centroid of the set of non-relevant documents and closer to the centroid of the set of relevant documents.

Using pseudo-relevance feedback to do query expansion is proven to be useful and produces a better query representation in most of the cases. However, in some cases, this method may perform badly because some new terms we add to the original query may be unrelated to the “optimal query” and harmful to the retrieval task. To overcome this issue, Cao, Guihong, et al. [5] proposed a method that uses machine-learning methods to select good expansion terms used for pseudo-relevance feedback.

In summary, using pseudo-relevance feedback to perform query expansion is a widely used method in retrieval systems and Rocchio algorithm is often used in vector space retrieval model to implement it. To overcome the issue of adding weights to unrelated expansion terms, we need a method to systematically select good expansion terms. In the next section, we state the contribution of this thesis on building the system that collects information from the web for entities as well as this particular problem on improving query expansion using pseudo-relevance feedback.

1.3 Contribution

In this thesis, we build a retrieval system that systematically collects information from the web in order to complete and update information about entities stored in the database. Our system

first connects to the database we currently have and load the existing data about the entities. Then it crawls the web by using this information from the database and retrieves a document pool. From this document pool, it systematically constructs queries that retrieve the initial set of documents. After that, it performs query expansion by using a novel pseudo-relevance feedback method and constructs new queries. By using new queries it retrieves new set of relevant document about entities from the document pool. Finally the system integrates the newly retrieved documents into the existing information and presents it as a whole to the users.

In the process of query expansion, we propose a novel approach of doing pseudo-relevance feedback based on the baseline Rocchio Algorithm. The method we use takes the existing attribute values about the entities stored in the database as input and treats this information as a special document set. We modify the formula that constructs the new query by boosting the terms that occur in this special document set. In this process, we make an assumption that for a given entity, the terms occur in the existing database are related to the relevant documents for this entity even though these terms may belong to other entities. Therefore by boosting these terms we may get a better query representation than the traditional query expansion.

The rest of the thesis is organized as follows. In Chapter 2, we overview and formulate the problem. In Chapter 3 we introduce the system we design by describing each step of the retrieval task. In Chapter 4, we discuss in detail the method we use for doing pseudo-relevance feedback by leveraging the existing information in the database. Then in Chapter 5, we describe how we evaluate our system in a specific case: collecting alumni information from the web to

complete and update the alumni database maintained by the College of Engineering of the University of Illinois. Finally we conclude and discuss future works in Chapter 6.

CHAPTER 2

PROBLEM OVERVIEW

In many applications we are maintaining a database that stores information about certain entities (e.g., a person or a product). In some cases the information we have is limited and we want to collect more data from the web about them.

We formulate this problem by defining the database first. Each entity is stored as one row in the database and each row may have many attributes. Each row is corresponding to one entity and each column is corresponding to one attribute. One of the attributes is some kind of identifier of the entities (e.g., the name of a person). These attributes may contain text values or numerical values. Some entities have complete information, meaning every attribute has a value while other entities are missing values in some attributes. For example, a person entity has attributes “name”, “title”, “company” and “university” indicating this person graduated from this university and works for this company with some title. This database is incomplete and in many rows we only know partial information about the “person” entities. For example, we only know a person’s name and the university he graduated from but have no information about his current employment.

The problem we want to solve is to collect more information about these entities from the web. The information we are collecting is in the form of web pages. Therefore, the problem is defined as follows. Given the database we described above, for each entity, we want to return a ranked list of web pages that are relevant to it. Relevance here means the web page is mainly

about that entity and contains important information about it. In the example we described above, relevant web pages about a person entity are his or her homepage, social network profile or some news articles that are mainly about this person and so on.

In summary, for each entity given its attributes stored in the database, we should return a ranked list of web documents sorted by relevance by using the information stored in the database as additional knowledge.

CHAPTER 3

SYSTEM DESIGN AND IMPLEMENTATION

The retrieval system we design performs the task in several steps. Firstly, it connects to the database and loads the data. It then crawls the web and retrieves a big document pool for all the entities. Then the system performs indexing and searching on this document pool to return a ranked list of web pages for each input entity. Finally it presents the retrieved web pages along with the information we already have as a whole to users. We start this section by describing how it crawls the web.

3.1 Web Crawling

In order to collect relevant data for the entities we first crawl the web by using some simple queries. By doing web crawling we get a big collection of data that can be used for further searching.

The system crawls the web by first constructing a query for each entity. Since each entity has an attribute that serves as an identifier, we include this value in the query. To make the document pool containing more accurate and relevant web pages, we also include one additional column in the database. This column is chosen for two reasons. First including this attribute value in the query narrows down the target document pool significantly. In other words, it eliminates great number of web pages that refer to some other non-relevant entities with the same identifies as our target entity (e.g., some other people with the same name as

our target person). Second reason is that this attribute in the database is non-empty for most of the entities so that we may include it in most of the queries.

After the query is constructed for each entity, the system uses it to crawl the web by querying Google. Google search API is used in our system. We set a limit number of web pages and store the returned web pages locally. All the web pages returned are stored together to become a document pool.

Basically web crawling is our first step of data collection. Relevant web pages (i.e., the desired output) are included in this document pool but most of the web pages in it are non-relevant. Therefore, further work needs to be done to select the most relevant web pages about the entities out of it.

3.2 Indexing and Searching

To further select relevant web pages from the document pool, the system indexes the documents and performs searching. We use Lucene [6], which is a widely used open source project for information retrieval. It provides Java implementation of many model information retrieval functionalities including indexing and searching. It also gives us the flexibility to customize our retrieval models.

3.2.1 Indexing

In the process of indexing, the system first transforms the raw html web pages into text documents by removing html tags and extracting the content. Then Lucene builds the index by treating each web page as a single document. In addition to storing term-document references, we also store the term-vectors [6] that contain information of every term frequencies and inverted document frequencies for each term. Term-vectors are mainly used later for query expansion. For each document, the system can easily retrieve the vector representation by using term-vectors. It is also useful for getting the top terms that are used to expand the query.

3.2.2 Searching

For each entity, the goal of searching is to retrieve the most relevant web pages (i.e., documents) out of the big document pool and return them as the final output of our system. Basically the input to our system is the unique identifier of the entity the user want to search. And the information we have is the database entry of that entity, which contains all the attribute values with some of them being empty. We may also leverage the entries of other entities in the database as additional knowledge in order to improve retrieval accuracy. Rather than the simple query we use in web crawling step, we need a systematical way to construct an optimal query that retrieves the most relevant documents.

The searching process we design contains several steps. Firstly, the system performs an initial search by using a query constructed by concatenating several attribute values of the input entity. These attributes we use to construct the query are chosen manually. They need to

be the attributes that best represent the identity of the input entity. In the case of searching alumni, we select person's name, the university he or she graduated from, the company he or she works for and the employment title. We discuss this case in detail in Chapter 5.

Since our database suffers from the problem of incompleteness, in many cases, some of the attribute values are empty. This problem hurts the accuracy of the results significantly. In order to overcome this issue, our system performs query expansion using pseudo-relevance feedback. By leveraging the information stored in the database as additional knowledge we propose a new way of doing pseudo-relevance feedback, which does the query expansion and returns a new query. More detail on how the system works to expand the query is discussed in the next chapter.

After the system constructs the new query, it performs the second searching and returns a newly ranked documents as the final output.

In both searching steps, the system uses the vector-space model [3] to assess relevance. Both queries and documents are represented as vectors where each dimension is corresponding to a term. The term weighting (i.e., the value of each dimension) is computed by using TF-IDF scheme [2] as follows:

$$term\ weighting = tf(t, d) \cdot idf(t)^2$$

Where:

- $tf(t, d)$ measures how frequent the term t appears in document d given how many times t appears in d , denoted as $freq(t, d)$:

$$tf(t, d) = \sqrt{freq(t, d)}$$

- And $idf(t)$ stands for inverse-document frequency, which measures how rare the term is in the whole collection:

$$idf(t) = 1 + \log \left(\frac{|D|}{docFreq(t) + 1} \right)$$

Where $|D|$ is the total number of documents and $docFreq(t)$ is the number of documents in which the term t appears.

Relevance of a certain document given a query in vector-space model is measured by computing the similarity between the two vectors. In our system, cosine similarity [3] is used to give every document a score given a query:

$$Cosine_Similarity(q, d) = \frac{V(q) \cdot V(d)}{|V(q)| |V(d)|}$$

Where $V(q)$ and $V(d)$ are weighted vectors for the query and the document, $V(q) \cdot V(d)$ is the dot product and $|V(q)|$ and $|V(d)|$ are their Euclidean norms.

Given the vector representation and scoring function described above the system ranks all the documents given a query and returns a ranked list based on the scores in both initial searching and final searching after query expansion using pseudo-relevance feedback.

3.3 Result Integration

For each entity, our goal is to collect relevant web pages that provide the users with more information in addition to the database with limited information. Our system achieves this goal by presenting an html page that integrates the returned web pages with the information we have in the database. Below is an example of this html page with the entity being a person.

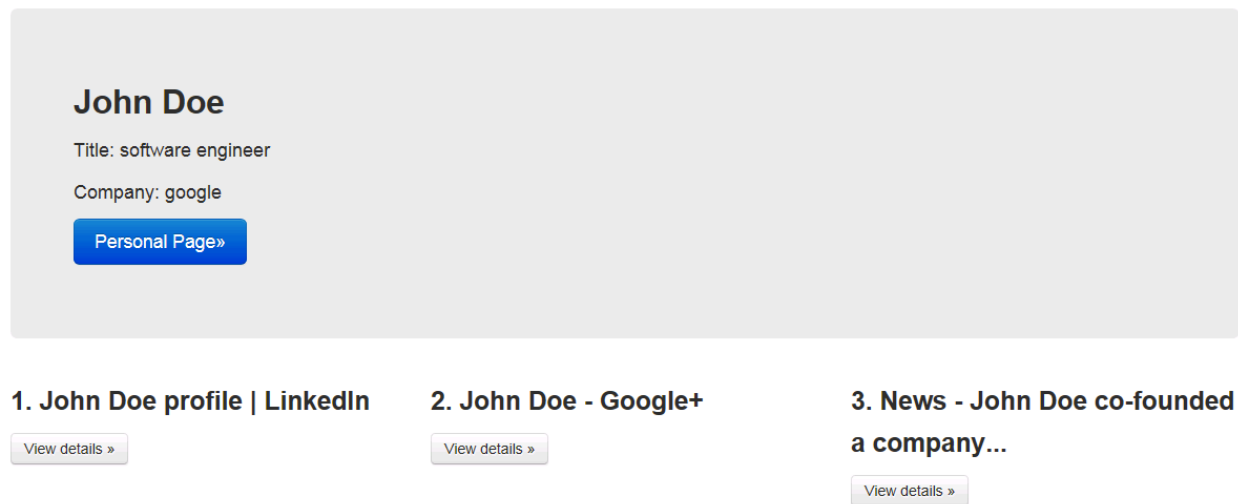


Figure 3.1: An example of the html page for a person our system generates.

The entity's name along with its attribute values from the database is printed first. Then the top 3 relevant web pages the system retrieves are listed. Only the titles of the web pages are printed and the user may choose to view detail by clicking the button below each of the page. The button leads to the actual html page. In this example, the entity is a person. In our database we have this person's name, employment title, company he works for and a link to his personal page. Our system retrieves 3 most relevant web pages. Two of them are social network profiles and the third one is a news article about this person. Note that in this example

a button that leads to this person's personal page is provided below his basic information but this is not the general case.

In summary, this chapter describes all the steps our system performs in order to retrieve relevant web pages about entities. In the next chapter we discuss in detail how we improve retrieval accuracy by implementing a novel pseudo-relevance feedback method in the searching process.

CHAPTER 4

PSEUDO-RELEVANCE FEEDBACK

In the process of searching relevant web pages we talk about in the previous chapter, our system uses pseudo-relevance feedback to expand the query after performing the initial search in order to improve retrieval accuracy. In this chapter, we discuss in detail how our system leverages the data stored in the database and performs pseudo-relevance feedback with selected terms boosted. This novel method achieves higher performance than the baseline pseudo-relevance feedback method based on our evaluation in Chapter 5.

Pseudo-relevance feedback [2] assumes that the most frequent terms in the top K retrieved documents are useful for the retrieval task. So the method expands the original query by adding those terms. However, in [5] Cao, Guihong, et al. claims that in reality this assumption doesn't hold all the time. Some of terms added to the query are actually not relevant and may be harmful to the retrieval. Therefore selecting only those terms that are relevant could be an important way to improve pseudo-relevance feedback performance and retrieval accuracy.

In our problem, since we have a database that serves as an additional knowledge, we make an assumption that the terms that appear in the database are more relevant to the retrieval task than others. In other words, among all the terms that are selected by traditional pseudo-relevance feedback method from the top K retrieved documents, we assume that those of them also appear in the database are more relevant and need to be boosted. Based on this

assumption our system modifies Rocchio Algorithm [4] by boosting those terms that appear in the database in the process of query expansion. Detail implementation is as follows.

Our system treats the database as a special document collection. In this collection each column is treated as a single document and every attribute value is tokenized into terms. We introduce selected term frequency $stf(t)$ and selected inverse document frequency $sidf(t)$ for each term t , which are defined as follows.

$$stf(t) = \sqrt{freq(t) \text{ in database}}$$

$$sidf(t) = 1 + \log \left(\frac{|C|}{docFreq(t) + 1} \right)$$

Where $|C|$ is the total number of columns and $docFreq(t)$ is the number of columns that contain term t . Note that naturally those terms that don't appear in the database have $stf(t)$ being zero.

Then we define our new query expansion formula as follows.

$$\overrightarrow{Q_{new}} = (a \cdot \overrightarrow{Q_o}) + \left(b \cdot \frac{1}{|D_r|} \cdot \sum_{\overrightarrow{D_j} \in D_r} \overrightarrow{D_j} \right) + (c \cdot \vec{S})$$

Where \vec{S} denotes selected term vector. In \vec{S} , each dimension corresponds to the same term as query vector and document vector and the value of each term t has a selected term weighting defined as:

$$selected \text{ term weighting} = stf(t) \cdot sidf(t)^2$$

Note that we discard the non-relevant document vector part since we only consider top K retrieved documents as relevant documents. Basically our system boosts those terms that appear in the database when we expand the query so that in the final query vector, these terms will have higher weights. Other terms that don't appear in the database don't get boosted since they have their $stf(t)$ being zero. And boosting factor depends on how important the term is in the database, which is measured using similar formula as TF-IDF.

CHAPTER 5

EVALUATION

To evaluate our retrieval system, we conduct experiments on a case of alumni web pages searching. We use the database maintained by the College of Engineering of the University of Illinois. Experiments show that our system can effectively retrieve relevant pages of alumni on the web and that the novel pseudo-relevance feedback method outperforms a simple baseline approach.

5.1 Dataset

The database from the College of Engineering of the University of Illinois contains information about alumni graduated from the University of Illinois. The original database has over 30 different attributes (i.e., columns) and over 8000 rows (i.e., alumni). For most of the rows, we don't have complete information. In other words, majority of the cells in our database are empty. This fits the situation we describe that the data we have is limited.

From this database we create a dataset for experiments in the following way. Among all the attributes, we select a subset of them, which is given below.

Table 5.1: Selected attributes in alumni database

| Name | University | Title | Company | Major | Degree Year |
|------|------------|-------|---------|-------|-------------|
|------|------------|-------|---------|-------|-------------|

Where “Company” is the most recent company this person works for and “Title” is the employment title in that company. The column “University” is added manually and every alumnus has the same value “University of Illinois”.

Then we select 112 alumni (i.e., rows) that have complete information (i.e., all of their attributes have non-empty values). From these 112 alumni data we construct a golden dataset that serves as testing purpose. The way the golden set is constructed is to query Google by using the combination of all the attribute values as queries. We collect top 3 web pages returned by Google for each person and assume that they are the most relevant web pages for this person. Thus, for each entity, the goal is to retrieve these 3 documents.

5.2 Experiments and Results

To evaluate our system and our pseudo-relevance feedback method, we need to show that given a database with limited information, our system can achieve higher accuracy than both the baseline algorithm and pseudo-relevance feedback method using Rocchio algorithm. Note that the baseline algorithm we use here is just using the combination of all the attribute values we have as queries.

In order to achieve this we define a retrieval task in the following way. We use the dataset with 112 alumni information described above and pretend that we only know limited information. Three different cases are experimented. We use precision at 3 retrieved documents and MAP (Mean Average Precision) as measures. Below are the results of

comparison among the baseline approach, Rocchio Algorithm and our method under three different cases:

Table 5.2: Experiment results in case 1: we only have information in “Name” column

| Method | Precision at 3 | MAP |
|-------------------|----------------|-------|
| Baseline | 0.424 | 0.512 |
| Rocchio Algorithm | 0.384 | 0.461 |
| Our system | 0.714 | 0.758 |

Table 5.3: Experiment results in case 2: we only have information in “Name” and “University” columns

| Method | Precision at 3 | MAP |
|-------------------|----------------|-------|
| Baseline | 0.513 | 0.584 |
| Rocchio Algorithm | 0.393 | 0.497 |
| Our system | 0.750 | 0.778 |

Table 5.4: Experiment results in case 3: we only have information in “Name” and “Major” columns

| Method | Precision at 3 | MAP |
|-------------------|----------------|-------|
| Baseline | 0.629 | 0.691 |
| Rocchio Algorithm | 0.509 | 0.581 |
| Our system | 0.629 | 0.674 |

From the results we see that the regular pseudo-relevance feedback doesn't improve retrieval accuracy in all the three cases. In the first two cases, our system outperforms both the baseline algorithm and Rocchio Algorithm significantly. Especially in the first case, in which we only know very little about our entities (i.e., we only know names) our method can still achieve a fairly high retrieval accuracy. In the third case, however, our system doesn't improve performance compared to the baseline algorithm.

To better understand the performance of our system, we conduct analysis on the terms that are added to the original query from the pseudo-relevance feedback documents. We compare the top terms used to expand the query from our system with those from Rocchio Algorithm in the retrieval task of a specific person. Result is shown in Table 5.5. Note that we replace the real name of that person with "John Doe". This person works for Google as a software engineer.

Table 5.5: Top 5 feedback terms that used to expand the query

| Rank | Rocchio algorithm | Our system |
|------|-------------------|------------|
| 1 | John | computer |
| 2 | Doe | John |
| 3 | kiong | engineer |
| 4 | conference | Doe |
| 5 | ieee | Google |

With selected terms boosted, our system added “computer”, “engineer” and “Google” to the original query. These terms are very relevant to this person and our system boosts them because they show up frequently in some other rows from our database even though we don’t know this person actually works for Google as an engineer. From this analysis, we show that our method performs better in query expansion using feedback than Rocchio algorithm.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we discuss finding relevant web pages for entities given a database with limited information. We propose and implement a retrieval system that performs this task effectively and introduce a novel pseudo-relevance feedback method that boosts relevant terms in the process of query expansion.

Our system crawls the web, indexes web pages and performs searching in order to retrieve relevant web pages for entities. In the process of searching, our system uses a novel pseudo-relevance feedback method with vector space implementation and expands the original query in order to improve retrieval accuracy. This novel method boosts selected terms that frequently appear in our database when performing query expansion.

We evaluate our system on the alumni dataset provided by the College of Engineering of the University of Illinois and show that our system achieves better performance compared to the traditional pseudo-relevance feedback method (i.e., Rocchio algorithm). More specifically, our system gets higher retrieval accuracy than both the baseline algorithm and Rocchio algorithm in the task of finding relevant web pages given a database with limited information. Our method boosts relevant terms that greatly help improving the retrieval accuracy.

In the future, we may explore more possibilities on improving the proposed method. In this thesis, we only perform experiments on vector space model and uses Rocchio algorithm as baseline. Many other retrieval models such as probabilistic model [7] may lead to better

performance. Also, only a simple formula similar to TF-IDF is used when selecting relevant terms from the database. Therefore, we may explore more measures in the future.

In addition, we may run our systems on more datasets. Since we only conduct experiments on the alumni database, it is very interesting to know how our proposed method would perform on other types of entities (e.g., product).

REFERENCES

- [1] Xu, Jinxi, and W. Bruce Croft. "Query expansion using local and global document analysis." *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1996.
- [2] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge: Cambridge university press, 2008.
- [3] G. Salton, A. Wong, C.S. Yang, A vector space model for automatic indexing, *Communications of the ACM*, v.18 n.11, p.613-620, Nov. 1975
- [4] Rocchio, Joseph John. "Relevance feedback in information retrieval." (1971): 313-323.
- [5] Cao, Guihong, et al. "Selecting good expansion terms for pseudo-relevance feedback." *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2008.
- [6] "Apache Lucene Core," 2013; <http://lucene.apache.org/core/>
- [7] Salton, Gerard, and Chris Buckley. "Improving retrieval performance by relevance feedback." *Readings in information retrieval* 24 (1997): 5.