Search Result Reranking Using Clustering

Stephen Yeboah

Master in Information Systems
Submission date: June 2011
Supervisor: Herindrasana Ramampiaro, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

In existing approaches to biomedical information retrieval, relevant documents according to the user are often at the bottom of the retrieved ranked documents (list). This may be as the result of users inability to formulate queries correctly and users not included in the ranking process. This reduces the retrieval performance i.e. recall and precision. The problem we try to address in this thesis is to find ways of fulfilling user information needs through statistical techniques (approaches).

Owning to the quality of information retrieval using the basic retrieval approaches on biomedical data (information), we believe it is possible to improve the retrieval performance to satisfy user information needs. The goal of this thesis is to test the possibility of using statistical approaches in Biomedical Information Retrieval (IR) to improve the retrieval performance. The statistical approach of emphasis in this thesis is the Expectation Maximization (EM) algorithm using the multinomial approach.

The Expectation Maximization (EM) was chosen because first, it works well with large document collection as it in the case of our dataset i.e. Text Retrieval Conference (TREC) MEDLINE dataset and second, it is good at finding incomplete data in order to accurately cluster relevant documents to satisfy users' information needs. The outcome of this algorithm on the biomedical data i.e. MEDLINE dataset and the result compared with the basic method and then the K-Means algorithm.

Submission Date:	9th June, 2011
Supervisor:	Heri Ramampiaro

Search Results Re-ranking Using Clustering

Stephen Yeboah

June 9, 2011

Supervisor: **Heri Ramampiaro, Associate Professor**

NTNU

Norwegian University of Science and Technology

Preface

This thesis is the final work on my masters degree, and marks the end of my study of Information System at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU).

Special thanks goes to my supervisor, Heri Ramampiaro, for his invaluable feedback, suggestions and ideas. I would also like to thank Olawande Daramola, Post Doctoral Fellow, for his assistance throughout the thesis. Finally, i would like to thank my family for their support and motivation throughout my studies.

Abstract

Information Retrieval is a research area that has gained attention over the past two decades. Few of these researches have taken place in the biomedical domain where satisfying users' information needs are relatively difficult to be met. The goal of this project is to find out if it is possible to use statistical methods in Biomedical Information Retrieval (IR) and improve retrieval performance, i.e. finding ways of fulfilling user information needs, in the biomedical domain using clustering with knowledge from the BioTracer project.

K-Mean and Expectation Maximization (EM) approaches to clustering have been implemented in this project with more emphasis on the EM. Both approaches are used to re-ranking users searched results in an attempt to find ways of fulfilling their information needs. Comparison between the Expectation Maximization and the K-mean are drawn in terms of their retrieval performance i.e. precision and recall, the performance of EM compared to existing approaches to search results re-ranking using clustering and problems faced while implementing the EM.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Information Retrieval is a research area that has gained attention over the past two decades. Few of these researches have taken place in the biomedical domain where satisfying users' information needs are relatively difficult.

The gradual increased in the amount of user needs for biomedical information has accounted for higher demands for biomedical information retrieval systems. According to [Ramampiaro, 2010], while this increase has helped researchers to be up-to-date, many existing systems tend to either have low precision i.e too broad or low recall i.e. too restrictive in terms of search results returned by the query.

The aim of this project is to investigate whether it is possible to use statistical methods like the Expectation Maximization (EM) in Biomedical Information Retrieval and thus improve retrieval performance i.e. recall and precision. The existing approaches for clustering like the K-mean are not very effective when there is an increase in the available dataset. This is where the EM algorithm is effective. It is an iterative method that assumes that the data were generated from models (clusters) and tries to discover these models from the available data.

Satisfying users' information needs in the biomedical domain are difficult to be met and these are some of the reasons outlined by [Ramampiaro, 2010]:

* *Difficulty faced in providing a unified method for both biomedical information indexing and retrieval because of domain-specific terms used in*

this domain.

* *Ambiguities as a result of the mixture of both biomedical specific terms and English terms.*

* *Lack of widely recognised terminology standards as people can come up with their own terms and change them at will.*

In this project, we investigate the possibility of using statistical methods like the Expectation Maximization in Information Retrieval(IR) to improve the retrieval performance i.e. recall and precision [Baeza-Yates et al., 1999a] .

1.2 Problem Definition

In existing approaches to biomedical information retrieval, relevant documents according to the user are often at the bottom of the retrieved ranked documents (list). This may be as the result of users inability to formulate queries correctly and users not included in the ranking process. This reduces the retrieval performance i.e. recall and precision. The problem we try to address in this thesis is to find ways of fulfilling user information needs through statistical techniques (approaches).

Owning to the quality of information retrieval using the basic retrieval approaches on biomedical data (information), we believe it is possible to improve the retrieval performance to satisfy user information needs. The goal of this thesis is to test the possibility of using statistical approaches in Biomedical Information Retrieval (IR) to improve the retrieval performance. The statistical approach of emphasis in this thesis is the Expectation Maximization (EM) algorithm using the multinomial approach.

The Expectation Maximization (EM) was chosen because first, it works well with large document collection as it in the case of our dataset i.e. Text Retrieval Conference (TREC) MEDLINE dataset and second, it is good at finding incomplete data in order to accurately cluster relevant documents to satisfy users' information needs. The outcome of this algorithm on the biomedical data i.e. MEDLINE dataset and the result compared with the basic method and then the K-Means algorithm.

1.3 Project Outline

Chapter 1 gives a brief introduction to this project, Chapter 2 introduces the concepts and definitions related to information retrieval that are used or applied in this thesis, tools and technologies learned tin this thesis. Chapter 3 gives some of the existing works that have been done in this field and Chapter 4 gives our implementation of the K-mean and EM approaches to searched result re-ranking using clustering. In Chapter 5, we evaluate our approach based on the results and comparison with the K–Mean and finally Chapter 6 provides the summary, concluding remarks and any suggestions for future work

Chapter 2

Preliminaries–Concepts and Definitions

2.1 Clustering

Clustering is the process of grouping document collections into subgroups or clusters based on either a set of internal or external properties or characteristics. Clustering is like classification except that there is no human actor to label the documents or assign document(s) to a particular class, label or group. Clustering is a typical example of unsupervised learning problem [Manning et al., 2008a] .

Clustering can be categorized into Hierarchical, Flat or Model-based. Hierarchical Clustering creates hierarchy of clusters with parent-child relationship. According to [Steinbach et al., 2000], the hierarchical approach to clustering is often seen as the better quality approach but its quadratic time complexity is seen as its major shortcoming. An example of this approach is the Agglomerative. Flat Clustering produces a flat set of subgroups without any clear-cut structures that will relate the groups to each other. As compared to the hierarchical, it produces inferior clusters but it thrives on its linear time complexity [Steinbach et al., 2000]. Example is the K- Means. The Model-based approach is based on the assumption that the data (documents) available were produced from models i.e. clusters and tries to produces these clusters from the data.

Another distinction can be made between hard and soft clustering. Hard Clustering computes hard assignment i.e. each document is assigned to one and only one cluster or group. On the other hand, Soft Clustering computes soft assignment – each document can be assigned to more than one group or cluster i.e. a document has a fractional membership in more than one cluster [Manning et al., 2008a].

2.2 Models

Retrieval model is an abstraction of the actual retrieval process. There are three (3) classic models namely Boolean, Vector and Probabilistic. In the boolean model, documents and queries are represented as a set of index terms. Documents and queries in the vector space model are represented as vectors in a high–dimensional space. Queries and documents representation in the probabilistic model are based on the probability theory.

Boolean retrieval model is the simplest model where documents and queries are represented as a set of index terms based on set theory and boolean algebra. In this model, all documents are either relevant or non-relevant as ranking is not supported. Query formulation is simple with precise meaning but sometimes requires experts to handle it. One major limitation of this model is, since partial matching is not supported, too many or too few documents are retrieved.

The vector model is the most common retrieval system. It recognizes the limitations of the boolean model i.e. binary document term weights, and proposes a structure that support partial matching. Everything in this model (documents, terms and queries) are represented as vectors in a high-dimensional space. This model supports ranking and term weighting and it is easy and fast to implement.

The probabilistic model captures the IR problems within a probability structure. It is theoretically "sound" but it is costly to implement as compared to the vector space model. This model requires an initial guess and is also based on the independence assumption which states that [Baeza-Yates et al., 1999d]:
Given a user query q and document d in a collection, the model tries to estimate the probability that the user will find the document d relevant.

2.3 Measuring Retrieval Performance

The two most widely known measures of retrieval performance are recall (completeness of retrieval) and precision (purity of retrieval). Recall and precision are defined based on the assumption: " *a binary relevance judgements, namely that every retrievable item or document is perceived as relevant or not relevant* "[Buckland and Gey, 1994].

Recall is defined as the portion or fraction of the relevant item or document that is retrieved where as precision is the fraction of the retrieved item that is relevant. Recall can also be the number of retrieved relevant documents as a fraction of all relevant documents. The number of retrieved relevant documents as a fraction of all retrieved documents also defines precision. An inverse relationship exist between recall and precision. As recall increases, precision decreases and vice-versa [Baeza-Yates et al., 1999c],[Buckland and Gey, 1994] .

Recall and precision can be illustrated with the help of Figure 2.3.
Let $|R|$ be the number of known relevant document
Let $|A|$ be the number of documents retrieved as a result of the query and $|Ra|$ be the number of relevant document that is retrieved. Therefore recall and precision can be defined mathematically as shown below.
$recall = |Ra|/|R|$
$precision = |Ra|/|A|$

2.3.1 Mean Average Precision (MAP)

Mean Average Precision (MAP), a primary evaluation measure for effective runs of all different runs submitted by researchers, has become popular in the TREC Community to improve retrieval performance in the area of genomic [Hersh et al., 2006][Johannsson, 2009][Manning et al., 2008b].

The idea behind MAP is to generate a single value (score) by averaging the precision figures obtained after each new relevant document is seen in the ranked list [Baeza-Yates et al., 1999c]. MAP is 0 (zero) if no relevant document is retrieved or seen in the ranked list. The MAP favours retrieval systems that retrieve relevant documents quickly and suffers poor performance in terms of the overall recall. As indicted by Johannsson [2009], measure of 0.4 can be caused by many factors with MAP, while the same R-precision value on a query having 10 relevant documents would mean 4 relevant doc-

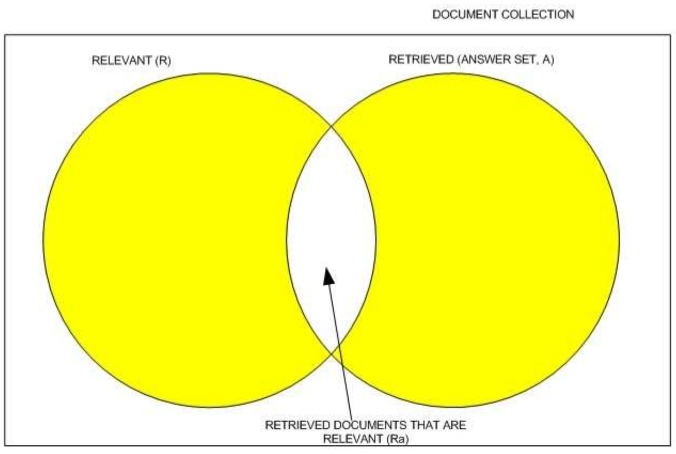
DOCUMENT COLLECTION
RELEVANT (R)
RETRIEVED (ANSWER SET, A)
RETRIEVED DOCUMENTS THAT ARE RELEVANT (Ra)

Figure 2.1: Illustration of recall and precision

uments were retrieved. MAP is defined as [Manning et al., 2008b]

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

where the set of relevant documents for query $q_j \in Q$ is $\{d_1, \ldots, d_{mj}\}$ and R_{jk} is the set of ranked retrieved documents down to document d_k.

2.4 Indexing

Indexing precedes searching. An index term is document or word whose meaning helps in recognizing the document's main theme. The purpose of indexing is to speed up the searching process or task. The nature i.e. size and dynamism, of an index (data structure built over the document/text collection) determines whether it is worthwhile to build and maintain an index [Baeza-Yates et al., 1999d] . There are several indexing techniques and the most notable ones are the Inverted Files or Index, Suffix Arrays and Trees and the Signature files. The inverted index is the most popular among the list while the suffix arrays and trees are better suited for phrase search [Baeza-Yates et al., 1999b].

The inverted index is "a word-oriented mechanism for indexing a text collection to speed up the searching task/process" [Baeza-Yates et al., 1999b]. It is made up of two parts: *the vocabulary and the occurrences.* The vocabulary is made up of the set of all the different words including numerals in the document or text collection after all the stop words have been removed. The occurrences stores the positions of these words or numerals in the vocabulary. Figure 2.2 shows an inverted index. In Figure 2.2, a line of text extracted from document, *d*, and the beginning of each from from the text is numbered. From this, an inverted file is constructed with the *vocabulary* and *occurrence*. Figure 2.3 shows inverted index over more than one document. Apache Lucene uses this approach in indexing documents. In Figure 2.3, three titles are extracted from three doucuments labeled 1, 2 and 3. The titles are index after the stop words have been eliminated.

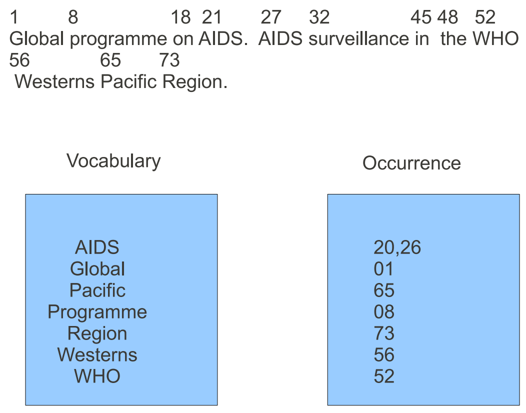

Figure 2.2: Inverted Index over one document

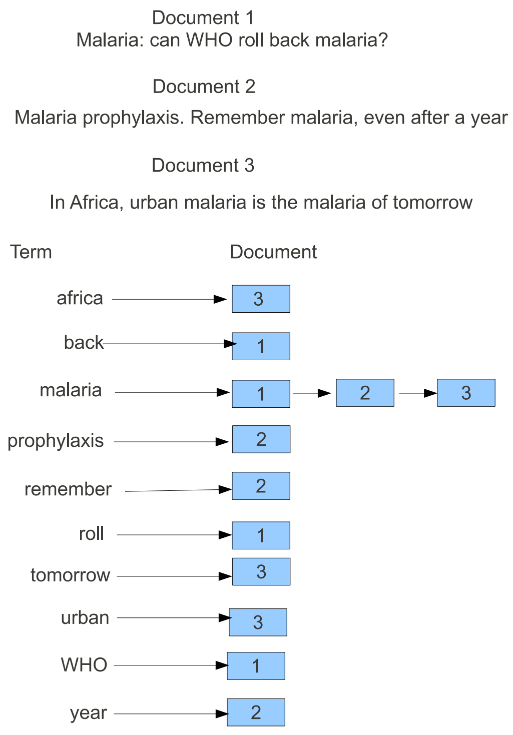
Document 1
Malaria: can WHO roll back malaria?
Document 2
Malaria prophylaxis. Remember malaria, even after a year
Document 3
In Africa, urban malaria is the malaria of tomorrow
Term
Document
africa
3
back
1
malaria
1
2
3
prophylaxis
2
remember
2
roll
1
tomorrow
3
urban
3
WHO
1
year
2

Figure 2.3: Indexing multiple documents

2.5 Lucene

In order to refrain from reinventing the wheel, our implementation took advantage of work done by Apache Lucene [1] for indexing and searching documents. Lucene is a high performance, full-featured and scalable Information Retrieval (IR) library or search engine written in Java [Hatcher et al., 2009]. Information retrieval refers to access to documents or information that satisfy user information needs from document collection. Lucene allows developers to incorporate searching capabilities to their applications ranging from databases to search engines. It is not full-searched program or application but just Application Programming Interface (API) that you can incorporate into your application. Lucene can index as well as make searchable any data that has been extracted from text or binary form [Hatcher et al., 2009].

Lucene is made up of 2 parts or components: *indexing and searching*. The indexing precedes the searching as the whole idea of indexing is to speed up the searching task. The indexing is a chain of steps including retrieving raw data (content) from file or any other repository, creating documents (set of fields) from the raw data and finally indexing these created documents. The searching process begins as soon as the indexing process is completed. It requires a user interface where users formulate queries, means of building the query programmatically, running the query to retrieve matched document(s) and finally displaying the results of the query to the user [Hatcher et al., 2009].

[1] http://lucene.apache.org/java/docs/index.html#Apache Lucene

Chapter 3

Related Work

Clustering search results is a research area that has gained attention over the past two decades. Many works have been done in the problem of clustering search results [Le][Zeng et al., 2004][Chen and Dumais, 2000][Ngo and Nguyen, 2005][Toda and Kataoka, 2005]. Most of these proposed solutions go about without the three(3) major components of traditional techniques: *Similarity function, threshold of similarity and pre–defined number of clusters. Similarity functions* are formed by considering different factors like the euclidean distance as in the case of K-Means, link structure in the case of hierarchical clustering (agglomerative). The *threshold of similarity* determines whether two(2) search results can be put into the same group or not [Le].

Arranging search results into clusters or groups allow users to easily and quickly browse through relevant search results. According to [Zeng et al., 2004], the traditional techniques suffer from one major limitation which is their inability to produce highly readable names. Normally, existing search engines sequentially display search results according to their relevance to the query. Users have to go through these results to find relevant ones among the lot which is time–consuming [Zeng et al., 2004][Toda and Kataoka, 2005].

Another reason for the attention received is the rise in information overload. Information overload occurs as a result of users' inability to formulate queries properly. This leads to too many search results returned by the search engine [Le][Zeng et al., 2004][Toda and Kataoka, 2005]. One approach of reducing this information overload problem as suggested by [Le][Zeng et al., 2004][Chen and Dumais, 2000][Ngo and Nguyen, 2005][Toda and Kataoka, 2005] is through clustering. Clustering these search results into groups allow the user to identify relevant group at a glance instead of moving through the search results sequentially as relevant documents tends to be similar to each other [Manning et al., 2008a][Zeng et al., 2004]. Again, organizing the search results into clusters enable users to concentrate on the documents in a particular cluster rather than browsing through the results one after the other in a sequential manner [Chen and Dumais, 2000].

The tremendous size of the web and low precision of user queries making it difficult to retrieve documents that satisfy user information needs has also contributed to the attention received over the years by this research area [Ngo and Nguyen, 2005]. Clustering the search results enables similar documents to be put together in one group. This simplifies the presentation of results in more compact form allowing users to quickly browse through the search result. According to [Toda and Kataoka, 2005], clustering of search results comes in two(2) flavours: *document–based* and *label–based*. *Document–based* clusters documents based on content similarity and the clusters produced do not overlap and names, not readable for users of the web to easily determine relevant documents from the search results. *Label–based* on the other hand extract information from the search result such as titles and phrases and produce cluster names that are readable for users to easily identify relevant documents from the search results [Ngo and Nguyen, 2005][Zeng et al., 2004][Le].

From the work of [Zeng et al., 2004], the solution proposed is summarized below:
Given the query and the ranked list of search results in the form of titles and snippets, their method extracts all possible phrases from the contents of the documents and calculate several properties for each phrase such as *phrase frequencies, edit distance and document frequencies.* A regression model learned from previous training data is then applied to combine these calculated properties into single significant score. The phrases are ranked according to this score. The names of the top salient phrases ranked according to this significant score becomes the names of candidate clusters which are further merged according to their corresponding documents.

[Le]'s solution has 2 main premises:

- User queries can be considered as a summary of returned search results
- The more specific the user query is, the fewer the results returned.

The summarized solution is as follows:

1. From the search results, a summary is generated or produced.

2. Results that match this summary are first clustered.
3. A summary is generated from the remaining search results.
4. Results that match this summary are clustered second.
5. Steps 3 and 4 are repeated until all results are clustered.

In the above approaches, the clustering process does not take the users' perspective, i.e which ones are relevant and which ones are not, into consideration. The result can therefore be rejected by the users since it might not meet their information needs. The difference between the above approaches and our approach is that, we assumed that the first ten (10) documents in the list are relevant and the remaining, non–relevant. the user is therefore somehow involved in the clustering process. The steps involved is as follows:

- The search result is displayed based on the query that the user formulated which may result in information overload or not.
- From the search result, a document is randomly selected from the first 10 documents regarded by the user as *relevant* and another one from the remaining documents regarded by the user as *non–relevant*.
- The search results are re–ranked based on the initial selected performed by the user resulting in relevant documents being placed above the list. We believe that this may improve the retrieval performance as the user was somehow involved in the clustering process.

Chapter 4

Reranking Using an EM Algorithm

Our aim in this project is to investigate the possibility of using statistical approaches (methods) in Information Retrieval (IR) to improve retrieval performance i.e. recall and precision. This chapter takes a looks at one of the statistical approaches namely the Expectation Maximization (EM) using the multinomial approach, briefly discuss the K-Means Algorithm and the implementation of both algorithms.

4.1 K–Means Algorithm

It is one of the most simplest and popular iterative flat clustering algorithm. The main goal or objective of this algorithm is to minimize the average square Euclidean distance of documents from the mean or center of their cluster, popular known as the centroid. The centroid, represented as $\vec{u}$, of the documents in a particular cluster, w can be determined from equation 4.1.

$$\vec{u}(w) = \frac{1}{|w|} \sum_{\vec{x} \in w} \vec{x} \tag{4.1}$$

where $\vec{u}$ = vector representing the mean of all the documents in a cluster, $|w|$ = number of documents in cluster, w, and $\vec{x}$ = vector representation of a document, d, in cluster, w [Krishna et al., 1999],[Manning et al., 2008a]. The K–Means alternate between two steps: *Reassignment* and *Recomputation*.

At the reassignment stage, the algorithm assign or reassign a document to a cluster with the minimum euclidean distance i.e. hard assignment, a document belongs to exactly one cluster. Once a document has been assigned to a particular cluster, the centroids of the clusters are recomputed at the re–computation step.

4.2 Expectation Maximization

The Expectation Maximization (EM) is a statistical approach or technique which is useful in various contexts ranging from standard incomplete–data problems to iteratively re-weighted least square and empirical Bayes models [Dempster et al., 1977], [Louis, 1982]. The "incomplete–data" situations according to [McLachlan and Krishnan, 2008], do not only include situations where it is evidently clear that there are missing data, truncated distributions or grouped observations but also in situations where the incompleteness is not evident or clear.

The EM, an example of the model–based clustering, is an iterative algorithm that assumes that the document or data available came from a model or cluster and tries to retrieve the original model from the data. The model(s) recovered from these data/documents define(s) the cluster(s) and the assignment of document/data to a cluster. Each iteration consists of two steps:*Expectation and Maximization* and thus the name, Expectation Maximization (EM) [Manning et al., 2008a] .

The main idea behind this algorithm according to [McLachlan and Krishnan, 2008] is: to connect or relate incomplete–data problems to complete–data problems for which the Maximum Likelihood (ML) estimation is computationally easier to manage or control. The algorithm also thrives in situations where other methods like the Newton–Rapson turns out to be complicated. The algorithm according to [Manning et al., 2008a] tries or seeks to find the parameters, Θ, that maximize the log-likelihood of generating data, $\mathcal{D}$, in equation 4.2 or 4.3

$$\Theta = argmax_{\Theta}\mathcal{L}(\mathcal{D}|\Theta) = argmax_{\Theta}log\prod_{n=1}^{N}\mathcal{P}(d_n|\Theta) \tag{4.2}$$

OR

$$\Theta = argmax_{\Theta}\mathcal{L}(\mathcal{D}|\Theta) = argmax_{\Theta}\sum_{n=1}^{N} log\mathcal{P}(d_n|\Theta) \tag{4.3}$$

where $\mathcal{L}(\mathcal{D}|\Theta)$ is the objective function the determines how good the clustering is. Given any number of clusters, we want to find the one that will maximize $\mathcal{L}(\mathcal{D}|\Theta)$.

Having determined the parameters, Θ, that maximize the likelihood of generating document according to equation 4.2 or 4.3, we compute an assignment probability, $\mathcal{P}(d|w_k;\Theta)$, for each document–cluster pair. This assignment defines the soft assignment i.e. a document has fractional distribution over several clusters. In our case, we had two clusters, *Relevant* and *Not–Relevant*. A document can have 0.5 membership in the *Relevant Cluster* and 0.5 membership in the *Not–Relevant Cluster*. $\mathcal{P}(d|w_k;\Theta)$ can be computed by the equation 4.4

$$\mathcal{P}(d|w_k;\Theta) = \left(\prod_{t_m \in d} q_{mk}\right)\left(\prod_{t_m \notin d}(1-q_{mk})\right) \tag{4.4}$$

where $\Theta = \{\Theta_1,\ldots,\Theta_K\}, \Theta_k = (\alpha_k, q_{1k},\ldots,q_{MK})$ and $q_{mk} = \mathcal{P}(U_m = 1|w_k)$ are the parameters of the EM algorithm. $q_{mk} = 1$ if the document, d, in cluster, k, contains the term, t_m, and 0 if otherwise. α_k is the prior probability that the document, d, belongs to cluster, k, if no additional information about d is given. In our case, for the first iteration since there were only two clusters, we assigned α_k to 0.5. k is the current cluster number and K is the total number of clusters. $\mathcal{P}(d|\Theta)$ from equation 4.2 and 4.3 can also be computed by the equation 4.5

$$\mathcal{P}(d|\Theta) = \sum_{k=1}^{K}\alpha_k\left(\prod_{t_m \in d} q_{mk}\right)\left(\prod_{t_m \notin d}(1-q_{mk})\right) \tag{4.5}$$

The EM Algorithm alternate between the *expectation step* and *maximization step* to determine the parameter, Θ, that maximizes $\mathcal{L}(\mathcal{D}|\Theta)$. Unlike the K-Means, it computes the maximization step corresponding to re-computation of centroids in K-Means before the expectation step which corresponding to the assignment of document to a cluster.
In the maximization step, we compute the conditional or lexical parameter,

q_{mk}, and the prior probability, α_k.

$$q_{mk} = \frac{\sum_{n=1}^{N} r_{nk} I\left(t_m \in d_n\right)}{\sum_{n=1}^{N} r_{nk}} \qquad \alpha_k = \frac{\sum_{n=1}^{N} r_{nk}}{N}$$

where N is the total number of documents and r_{nk} is the soft assignment. In the expectation step, we calculate the soft assignment, r_{nk} used in the maximization step above.

$$r_{nk} = \frac{\alpha_k \left(\prod_{t_m \in d_n} q_{mk}\right)\left(\prod_{t_m \notin d_n} (1 - q_{mk})\right)}{\sum_{k=1}^{K} \alpha_k \left(\prod_{t_m \in d_n} q_{mk}\right)\left(\prod_{t_m \notin d_n} (1 - q_{mk})\right)}$$

[Manning et al., 2008a] . Figure 4.1 shows the EM algorithm

4.3 Implementation

The main aim of this project is to investigate the possibility of statistical approach in information retrieval and its influence on the retrieval performance. As a result, we implemented an application that uses Expectation Maximization algorithm, which is one of these statistical approaches, and compare its retrieval performance with that of the K–Means and the baseline i.e. using just Lucene to retrieve documents based on their relevance to the user query. We formulated 50 queries according to Text Retrieval Conference [1] (TREC) 2005 ad hoc topics and retrieved documents that were relevant to these queries. The search results are clustered using both the Expectation Maximization (EM) and the K–Means algorithms after the user has selected two documents from the search results: *one relevant and one non–relevant according to the user.*

4.3.1 Architecture Overview

Figure 4.2 shows the architecture of our system. The user formulates queries and decides to re–rank the search results with either the EM or K–Means

[1]http://ir.ohsu.edu/genomics/

```
PerformEMClustering(K,Vocabulary,documentCollection,selectedIndices)
  N ← NUMBER_OF_TERMS(Vocabulary)
  NumDoc ← NumOfDocuments(documentCollection)

  while(iteration is less than the threshold)
    Maximization step:
      For k ← 1 to K
        Foreach term in Vocabulary
          SumQ=0, SumR=0
          For j ← 1 to NumDoc
              SumQ=SumQ + $r_{term,j}$ * calculateTermFreq(term, j)
              SumR =SumR + $r_{term,j}$
          EndFor
          $q_{term,k}$ = SumQ/SumR
        EndFor
        If(first iteration) then
          $\alpha_k$= 0.5
        Else
          $\alpha_k$ = summation of $r_{term,j}$
        Endif
      EndFor
    Expectation step:
      For n ← 1 to NumDoc
        For k ← 1 to K
          Num=1, Den=0
          Foreach term in Vocabulary
            Num = Num * calFreq(n,k,term,q,α)
          EndFor
          Num = Num * $\alpha_k$
          For j ← 1 to K
             InnDen=0
             Foreach term in Vocabulary
               InnDen = InnDen * calFreq(n,k,term,q,α)
             EndFor
             InnDen=InnDen * $\alpha_k$
             Den = Den + InnDen
          EndFor
          R term, k= Num/Den
        EndFor
          Return r
      EndWhile
```

Figure 4.1: EM algorithm

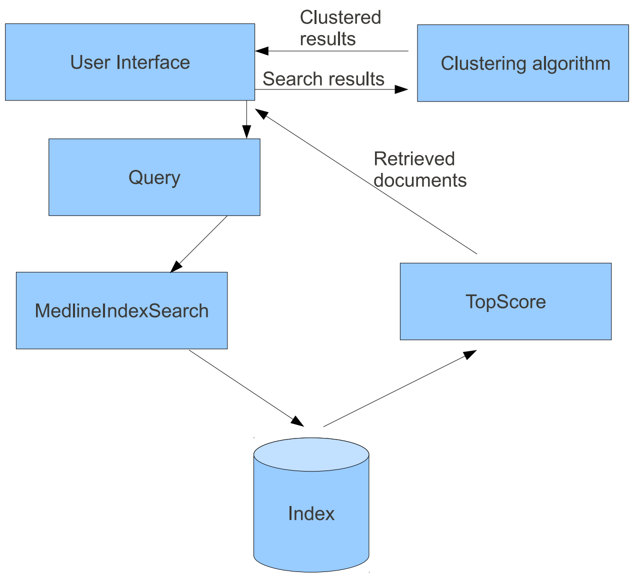

Figure 4.2: Architecture Overview

algorithm for the User Interface (UI). The query formulated by the user is accepted by the MedlineIndexSearch which retrieves documents that match the query. The TopScore provides a means to access these documents which are displayed on the UI for the user to view. From the search results, the user select two (2) documents: *one being relevant and the other, non–relevant* and decides to cluster the search results with either the EM or the K–Means algorithm to improve the retrieval performance i.e. *recall and precisoin.*

4.3.2 Document Indexing

As indicted in section 2.4, the purpose of indexing is to speed up tthe searching process. The outcome of the indexing process is *an index*: a word–based data structure. The indexing process in Lucene involves a number of steps as indicted in Figure 4.3

Data Collection

From Figure 4.3, our application heavily relies on the TREC 2004 dataset. In order to evaluate our investigations, a relevance judged document collection with biomedical domain is preferred. A set of relevance judgements is a list of relevant documents that experts have labelled as "correct solutions" for a given topic(s) [Johannsson, 2009]. This gives us a criteria or benchmark to evaluate search results using our approach.

Extract Text

The text from the TREC dataset was extracted with the help of classes from *java.io* namespaces notably *BufferedReader, FileReader and File.* Figure 4.4 shows snippets of the code used to traverse the dataset and extract the needed content (text).

Build Document

After we have extracted text from the dataset, we build documents to be used by our application. Usually, each document consists of name–value pairs called *fields.* In our case, each document consists of *PMID, Title, Abstract Text and Author Name.*

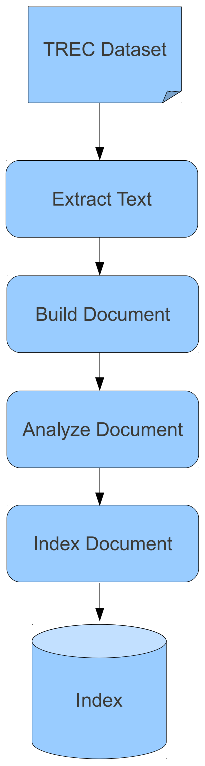

Figure 4.3: The Indexing process

```
     reader=new BufferedReader(new FileReader(file));
                String line=null;
                int counter=0;
                while((line=reader.readLine())!=null)
                {
                     separateTextFiles(line,counter);
                     counter++;

                }
   if(content.get(id).startsWith("PMID"))
             {
                 if(pmID == null)
                 {
                     pmID=new StringBuffer(content.get(id));
                     title=new StringBuffer();
                     abstractText=new StringBuffer();
                     authorName=new StringBuffer();
                     counter=0;
                     titleID=0;
                     authorID=0;
                 }
 else if(content.get(id).startsWith("SO"))
           {
                if(abstractText!=null)
                {
                  addDcoumentToIndex(spiltLine(pmID), spiltLine(title)
spiltLine(abstractText),spiltLine(authorName));
                }
                else
                {
                  addDcoumentToIndex(spiltLine(pmID),
spiltLine(title),spiltLine(authorName));
                }
```

Figure 4.4: some code snippets for extract text from the TREC 2004 dataset

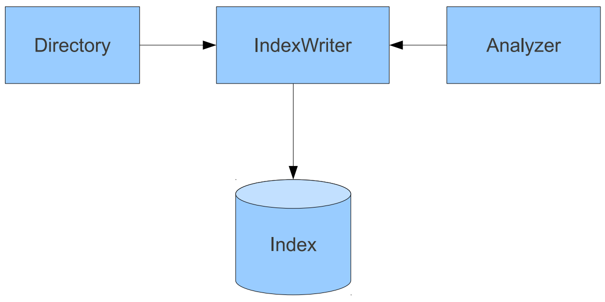

Figure 4.5: Lcene API's for indexing document

Analyse Document

Applications i.e. search engines do not directly index texts, but rather texts are broken into smaller units called *tokens*. At this stage, *stop words* i.e. words whose meaning are not significant enough to represent a document, are removed.

Index Document

This is the stage where documents are added to the *index*. Figure 4.5 shows Lucene API's that this stage make use of. From Figure 4.5, the Directory is where the documents will be indexed, the Analyzer is used for breaking texts in tokens, removing stop words and stemming and lemmatization and the IndexWriter does the adding of documents to the index stored by the Directory.

4.3.3 Implementation Details

The Text Retrieval Conference (TREC) MEDLINE citations are available in either 2 American Standard Code for Information Interchange (ASCII [2]) files approximately 2.6GB, a total size of 8.9GB or 5 Extensible Markup Language (XML [3]) files approximately 2.8GB, a total size of 19.2GB. For the purpose of our investigation, we chose to traverse the ASCII files. We identified documents within the citations using their identifiers[4].

The MEDLINE Citations were parsed with the help of Java classes from *java.io* namespace. As each document with the Citation is identified by its *PMID*, we used its *SO* as the delimiter. While parsing the Citations, we added a document to the index using Lucene whenever when encountered *SO* at the beginning of a new line as shown in the code snippets in Figure 4.6.

[2]http://en.wikipedia.org/wiki/ASCII

[3]www.w3.org/XML/

[4]Documents are identified in the judgement file using *PMID* (PubMed Identifier). Each PubMed Citation has a unique *PMID* number [Johannsson, 2009]

```
  else if(content.get(id).startsWith("SO"))
            {
                 if(abstractText!=null)
                 {
                   addDcoumentToIndex(spiltLine(pmID), spiltLine(title),
  spiltLine(abstractText),spiltLine(authorName));
                 }
                 else
                 {
                   addDcoumentToIndex(spiltLine(pmID),
  spiltLine(title),spiltLine(authorName));
                 }
                pmID=null;
                title=null;
                abstractText=null;
                authorName=null;
            }

private void addDcoumentToIndex(String pmid,String title,String
text,String authorName) throws CorruptIndexException,IOException
    {
        Document doc=new Document();
        doc.add(new
Field("PMID",pmid,Field.Store.YES,Field.Index.NOT_ANALYZED));
        doc.add(new
Field("TITLE",title,Field.Store.YES,Field.Index.NOT_ANALYZED));
        doc.add(new Field("CONTENT",title+" "+text+"
"+authorName,Field.Store.NO,Field.Index.ANALYZED,Field.TermVect
or.YES));
        medLineIndexer.addDocument(doc);
    }
```

Figure 4.6: Parsing the MEDLINE CItations

Chapter 5

Evaluation

The TREC Genomic Track test collection for 2004 is the document collection type chosen for evaluating our approach. It is available for the purpose of research in *biomedicine* and this suits our needs. The 2004 collection is a test collection with over 4.5 million MEDLINE citations. Out of these citations, 42,225 have been judged as relevant against 50 topics (47 out of these 50 topics were used as the remaining 3 had no *definite relevant* (DR) documents), each topic consisting of *ID, title, information needs* and *context* [Johannsson, 2009][Hersh et al., 2004].

5.1 Test Environment

The specifications for the platform used in our experiment is listed in table 5.1 .

Table 5.1: Test Environment

	Hardware Specification and Java Environment
CPU	Pentium (R) Dual–Core CPU T4200 @ 2.00Ghz
Memory	3GB
Operating System	Ubuntu 10.04 (lucid)
Storage	320GB
Java	Java (TM) SE Runtime Environment (build 1.6.0_24–b07)
Development Environment	Netbeans[1] (6.9.1)

5.2 Topics and Judgements

TREC Genomic Track has defined 50 different topics from which custom queries are formulated to search their document collection. These topics are categorized into 5 templates with each template containing 10 topics. The templates contain information that describes the following:

1. *standard methods or protocols for doing some sort of experiment or procedure.*
2. *the role(s) of a gene involved in a disease.*
3. *the role of a gene in a specific biological process.*
4. *interactions between two or more genes in the function of an organ or in a disease.*
5. *one or more mutations of a given gene and its biological impact or role.*

A document can be judged as not relevant (NR), possibly relevant (PR) and definitely relevant (DR). A document is seen as relevant if it is either possibly relevant (PR) or definitely relevant (DR).

5.3 Query Set

We created custom queries from the 50 topics provided by TREC Genomic Track to evaluate our approach. We implemented our queries based on Lucene's [2] *Query Parser* which support term, phrase and boolean queries. The following queries can be formulated for the topic: " *Provide information on the role of the gene BARD1 in the process of BRCA1 regulation"*

- BARD1 BRCA1 regulation :*Term query*
- BARD1 "BRCA1 regulation" :*Both term and phrase queries*
- +BARD1 +"BRCA1 regulation" :*Boolean query*–Both the term *BARD1* and the phrase *"BRCA1 regulation"* must appear in the document.

Each of these queries can give us different number of relevant documents. Appendix A contains the complete query sets.

[2]http://lucene.apache.org/java/docs/index.html

Table 5.2: Evaluation of the various approaches

Approach	MAP	R–Precision	P(10)	P(100)
Baseline	0.6284	0.6284	0.8136	0.1184
K–Means	0.6284	0.6284	0.8136	0.1184
Expectation Maximization	0.6284	0.6284	0.8136	0.1184

5.4 Evaluation Method

We used *trec_eval*[3] to evaluate our approach to determine whether it improves the retrieval performance. The *trec_eval* tool gives us a common ground to compare different Information Retrieval (IR) techniques or approaches. This gives us a basis to compare our result using both the Expectation Maximization (EM) Algorithm and the K–Means with researches already carried out in the TREC Community. The *trec_eval* is the standard tool used by the TREC community for evaluating an ad hoc retrieval run. It evaluates results based on the Mean Average Precision (MAP). It requires two files as input : *the topic–document output* and *the relevant judgement*. We created these files for the basic retrieval system (just based on the Java Lucene), the Expectation Maximization Algorithm and the K–Means for comparison.

5.5 Result

We submitted at most 20 documents for each query for both the basic IR and the retrieval system with the K–Means as well as for the retrieval system with the Expectation Maximization (EM). The number of documents for the EM is as a result of time constraint. The Expectation Maximization Algorithm using the multinomial approach considers each term in a document after it has been processed and this can be time–consuming for large document set. Table 5.2 shows a summary of the evaluation of various approaches, i.e. basic IR, K–Means and EM.

Our measurement is based on the first top 20 documents retrieved by a query instead of the default 1000 documents required by the *trec_eval* tool. According to [Spink et al., 2002] [Spink et al., 2001], a user is likely to look at documents on top of the ranked list so this may have affected our measure. 20 documents instead of the default value for the *trec_eval* tool was used due to

[3]can be downloaded at trec.nist.gov/trec_eval/index.html

the requirements of our test environment as shown in Table 5.1. We focused on the MAP as the evaluation measure for our approach since the *trec_eval* tool is good at MAP (which is the most standard measure of evaluation in the TREC Community) [Manning et al., 2008b].

From Table 5.2, the Mean Average Precision (MAP) and R–Precision are the same for all the approaches as well as the same precision after 10 and 100 documents have been retrieved i.e. P(10) and P(100).

Figures 5.1 and 5.2 a graph of Mean Average Precision (MAP) against Queries (Topics [4]) for the Baseline and K–Means and Expectation Maximization respectively. Figure 5.3 represents the comparison of Figures 5.1 and 5.2

5.5.1 Precision of top ranked results

In evaluating precision for the top ranked documents, we used the precision–at–k measure. According to [Johannsson, 2009], the precision–at–k measures how relevant the first k documents are to a query. This measure attempts to improve users satisfaction taking into consideration their behaviour i.e. most users rarely look past the top 10 documents in the ranked list [Spink et al., 2002][Spink et al., 2001][Baeza-Yates et al., 1999e]. Figure 5.4 shows precision after a specific number of documents retrieved for the various techniques used in this thesis.

5.5.2 Interpretation of Results

From Figure 5.3, the Expectation Maximization produces the same MAP value for all the queries as that of both the Baseline and the K–Means. Therefore our evaluation shows that the Expectation Maximization does not have effect on the precision.

5.6 Discussion

Our evaluation was based on the assumption that relevant documents according users, are found of top of the ranked list [Spink et al., 2002][Spink et al.,

[4]47 of the 50 topics were used as topics 18, 19 and 31 did not produced any relevant documents

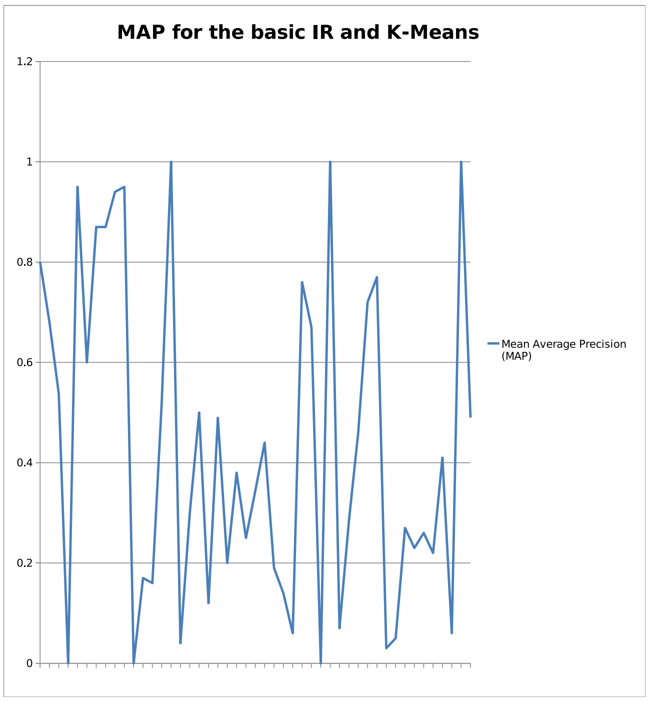

Figure 5.1: A graph of MAP against queries for both the Baseline and K–Means

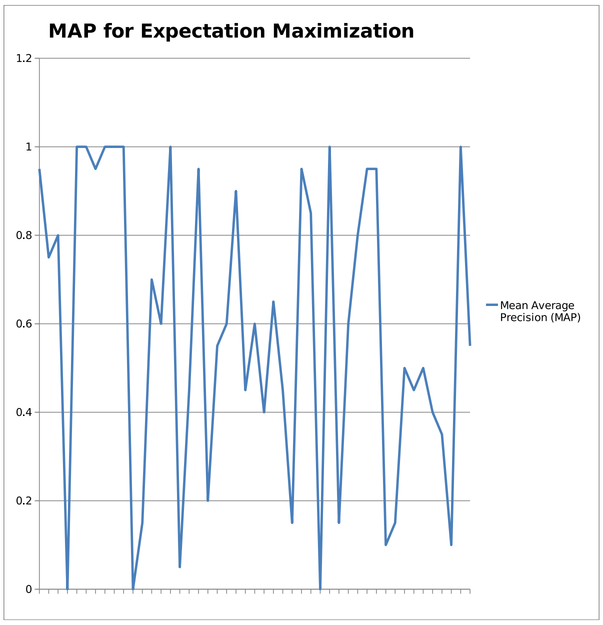

Figure 5.2: A graph of MAP against queries for the Expectation Maximization

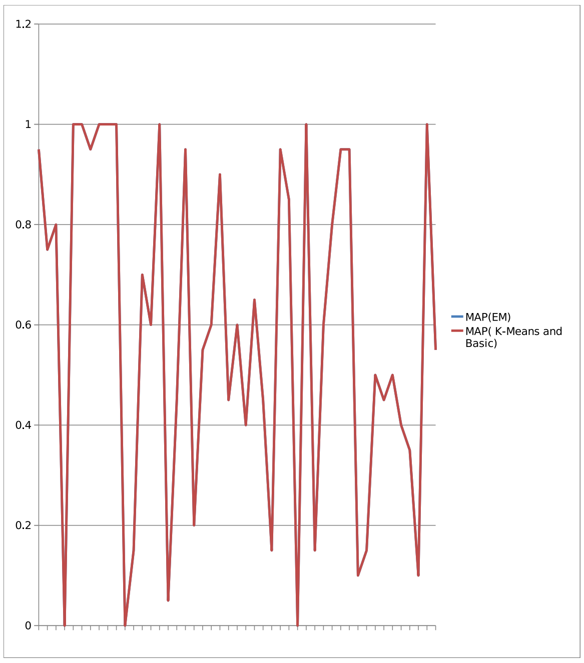

Figure 5.3: Comparison of Figures 5.1 and 5.2

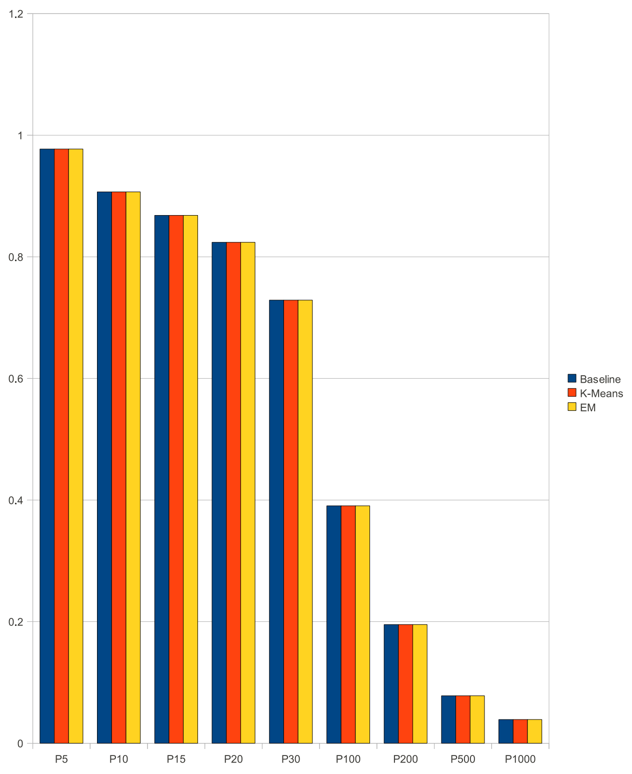

Figure 5.4: Precision after a specific number of documents retrieved

2001]. Hence, the first 20 documents for both the Baseline and K–Means, and the Expectation Maximization (EM) were submitted to the *trec_eval* tool instead of the 1000 documents required. 20 instead of 1000 documents for the EM was as a result of time constraint. This was owned to the fact that the EM considers every term in the documents retrieved after they have been preprocessed which can be time–consuming. This might have afftected the resulted MAP for the EM being the same as that of the other 2 techniques as indicted Table 5.2.

Our prototype implemented its queries based on *Lucene QueryPaser*, which supports all kinds of query type i.e. boolean, phrase, term, etc. We believe that has the query been extended to suit solely the biomedical domain, it could have further improve the retrieval performance. We made an assumption that our custom queries from the 50 topics represented real user queries. This might have improved the precision since we have experience as compared to most user in query formulation.

The document collection used for our evaluation is a *closed test collection* thus may not necessarily represent real world situations. A more proper evaluation according to [Johannsson, 2009] is to perform the evaluation on a large document collection where a human expert may be involved.

Chapter 6

Conclusion

6.1 Conclusion

In this thesis, we sought to find out the possibility of using statistical techniques in biomedical Information Retrieval(IR) and thus improve retrieval performance. In doing this, we implemented retrieval system prototype to test whether it was possible. We formulated 50 queries from the 50 topics provided by the TREC Community but only 47 of these were used as the remaining 3 produced no relevant document. We compared the statistical technique, i.e. the Expectation Maximization Algorithm, with the Baseline i.e. basic IR system and the retrieval system with the K–Means Algorithm based on the Mean Average Precision (MAP).

Using *trec_eval* for our evaluation, we found out that the Expectation Maximization (EM) does not have any effect on the MAP, i.e. generate the same MAP values as that of the K–Means and the Baseline. We also found out that the EM had the same precision after a specific number of documents have been retrieved compared to its counterparts i.e. the Baseline and K–Means.

6.2 Further Work

Our evaluation made use of 20 documents for both the baseline and K–Means, and the Expectation Maximization (EM) respectively. This we believe accounted for the same MAP values for the EM as that of its counterparts. We suggest that future works should extend the EM to 100 documents and if possible 1000 documents to have a fairer evaluation of all techniques used in this thesis.

Appendix A

Queries

A.1 Queries

1. +(method protocol) “open-up” +cell +electroporation
2. +(method protocol) “glutathione S–transferase” +cleavage “affinity chromatography”
3. +(method protocol) +“different quantities” +“different components” pour gel porous
4. +(method protocol) +“Green Flourescent Protein” tag proteins experiment
5. +(method protocol) +microsomal budding assay vesicles +microsomes vitro
6. +(method protocol) purification rat +IgM
7. +(method protocol) “Chromatin IP” isolate +proteins +DNA precipitate
8. +(method protocol) +Normalization microarray data
9. +(method protocol) +vivo ”protein–protein” time space cell
10. +(method protocol) standard +fluorogenic 5’-nuclease assay
11. +“Interferon–beat” +”Multiple Sclerosis”

12. PRNP “Mad Cow Disease”
13. “IDE” “Alzheimer’s Disease”
14. +MMS2 Cancer
15. +APC*adenomatouspolyposiscoli* +“Colon Cancer”
16. “Nurr–77” “Parkinson’s Disease”
17. +“Insulin receptor gene” Cancer
18. “Aapolipoprotein E” “Alzheimer’s Disease”
19. “Transforming growth factor-beta 1” “Cerebral Amyloid Angiopathy”
20. +“GSTM1” “Breast Cancer”
21. “nucleoside diphosphate kinase$NM23$” “tumor progression”
22. BARD1 “BRCA1 regulation”
23. “APC*adenomatouspolyposiscoli*” +“actin assembly”
24. COP2 CFTR +“endoplasmic reticulum”
25. “casein kinase II” “ribosome assembly”
26. “Nurr–77” +“auto–immunity” “T–cells” “lymph nodes”
27. +P53 +apoptosis
28. “alpha7 nicotinic receptor subunit” “ethanol metabolism”
29. “gamma-aminobrutyric acid receptors” “inhibitory synaptic transmission”
30. “Interferon–beta” “viral entry cell”
31. “BRCA1 regulation” ubiquitin cancer
32. L1 L2 “HPV11 Virus” “viral capsid”
33. +APC +“Colon Cancer”
34. +“phosphilipase A2” SAR1 “endoplasmic reticulum” vesicle “budding ER”

35. +CFTR +Sec61 degradation cystic fibrosis
36. “Bop Pes” “cell growth”
37. +“alpha7 nicotinic receptor” ApoE neurotoxic ethanol
38. “insulin-like GF” +“insulin receptor” function skin
39. HHF4 COUP-TFI +suppression +function +liver
40. Ret GDNF “kidney development”
41. “BRCA1 185de1AG mutation” “ovarian cancer”
42. Huntington “Huntington’s Disease”
43. “Sonic hedgehog” “developmental disorders”
44. NM23 “tracheal development”
45. +metazoan Pes “cell growth”
46. “hypocretin receptor 2” narcolepsy
47. +“presenilin–1” “Alzheimer’s Disease”
48. “alpha7 nAChR gene” alcoholism
49. FHM1 neuronal Ca2+ influx hippocampal neurons
50. “alpha 4-GABBA receptor” +impact +behaviour

“+” means *AND* meaning, the term(s) must appear in the documents that satisfy the given query. A term without the “+” sign means that a document may or may not contain the term.

Appendix B

Summary of our approaches using *trec_eval* tool

The results obtained by running trec_eval (summary output only) are

num_ret	Total number of documents retrieved over all queries
num_rel	Total number of relevant documents over all queries
num_rel_ret	Total number of relevant documents retrieved over all queries
map	Mean Average Precision (MAP)
gm_ap	Average Precision. Geometric Mean, q_score=log(MAX(map,.00001))
R-prec	R-Precision (Precision after R (= num-rel for topic) documents retrieved)
bpref	Binary Preference, top R judged nonrel
recip_rank	Reciprical rank of top relevant document
ircl_prn.0.00	Interpolated Recall - Precision Averages at 0.00 recall
ircl_prn.0.10	Interpolated Recall - Precision Averages at 0.10 recall
ircl_prn.0.20	Interpolated Recall - Precision Averages at 0.20 recall
ircl_prn.0.30	Interpolated Recall - Precision Averages at 0.30 recall
ircl_prn.0.40	Interpolated Recall - Precision Averages at 0.40 recall
ircl_prn.0.50	Interpolated Recall - Precision Averages at 0.50 recall
ircl_prn.0.60	Interpolated Recall - Precision Averages at 0.60 recall
ircl_prn.0.70	Interpolated Recall - Precision Averages at 0.70 recall
ircl_prn.0.80	Interpolated Recall - Precision Averages at 0.80 recall
ircl_prn.0.90	Interpolated Recall - Precision Averages at 0.90 recall
ircl_prn.1.00	Interpolated Recall - Precision Averages at 1.00 recall
P5	Precision after 5 docs retrieved
P10	Precision after 10 docs retrieved
P15	Precision after 15 docs retrieved
P20	Precision after 20 docs retrieved
P30	Precision after 30 docs retrieved
P100	Precision after 100 docs retrieved
P200	Precision after 200 docs retrieved
P500	Precision after 500 docs retrieved
P1000	Precision after 1000 docs retrieved

Table B.1: Evaluation Summary for the Baseline

runid	all	Baseline_Tag
num_q	all	44
num_ret	all	521
num_rel	all	848
num_rel_ret	all	521
map	all	0.6284
gm_map	all	0.5103
Rprec	all	0.6284
bpref	all	0.6284
recip_rank	all	1.0000
iprec_at_recall_0.00	all	1.0000
iprec_at_recall_0.10	all	0.9773
iprec_at_recall_0.20	all	0.8409
iprec_at_recall_0.30	all	0.8182
iprec_at_recall_0.40	all	0.7955
iprec_at_recall_0.50	all	0.6591
iprec_at_recall_0.60	all	0.5682
iprec_at_recall_0.70	all	0.4545
iprec_at_recall_0.80	all	0.4091
iprec_at_recall_0.90	all	0.3409
iprec_at_recall_1.00	all	0.1818
P_5	all	0.9045
P_10	all	0.8136
P_15	all	0.6970
P_20	all	0.5920
P_30	all	0.3947
P_100	all	0.1184
P_200	all	0.0592
P_500	all	0.0237
P_1000	all	0.0118

Table B.2: Evaluation Summary for K–Means

runid	all	K–Means_Tag
num_q	all	44
num_ret	all	521
num_rel	all	848
num_rel_ret	all	521
map	all	0.6284
gm_map	all	0.5103
Rprec	all	0.6284
bpref	all	0.6284
recip_rank	all	1.0000
iprec_at_recall_0.00	all	1.0000
iprec_at_recall_0.10	all	0.9773
iprec_at_recall_0.20	all	0.8409
iprec_at_recall_0.30	all	0.8182
iprec_at_recall_0.40	all	0.7955
iprec_at_recall_0.50	all	0.6591
iprec_at_recall_0.60	all	0.5682
iprec_at_recall_0.70	all	0.4545
iprec_at_recall_0.80	all	0.4091
iprec_at_recall_0.90	all	0.3409
iprec_at_recall_1.00	all	0.1818
P_5	all	0.9045
P_10	all	0.8136
P_15	all	0.6970
P_20	all	0.5920
P_30	all	0.3947
P_100	all	0.1184
P_200	all	0.0592
P_500	all	0.0237
P_1000	all	0.0118

Table B.3: Evaluation Summary for Expectation Maximization (EM)

runid	all	EM_Tag
num_q	all	44
num_ret	all	521
num_rel	all	848
num_rel_ret	all	521
map	all	0.6284
gm_map	all	0.5103
Rprec	all	0.6284
bpref	all	0.6284
recip_rank	all	1.0000
iprec_at_recall_0.00	all	1.0000
iprec_at_recall_0.10	all	0.9773
iprec_at_recall_0.20	all	0.8409
iprec_at_recall_0.30	all	0.8182
iprec_at_recall_0.40	all	0.7955
iprec_at_recall_0.50	all	0.6591
iprec_at_recall_0.60	all	0.5682
iprec_at_recall_0.70	all	0.4545
iprec_at_recall_0.80	all	0.4091
iprec_at_recall_0.90	all	0.3409
iprec_at_recall_1.00	all	0.1818
P_5	all	0.9045
P_10	all	0.8136
P_15	all	0.6970
P_20	all	0.5920
P_30	all	0.3947
P_100	all	0.1184
P_200	all	0.0592
P_500	all	0.0237
P_1000	all	0.0118

Bibliography

R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern Information Retrieval.* ACM press New York, 1999a.

R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern Information Retrieval, Chapter 8*, volume 463. ACM press New York, 1999b.

R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern Information Retrieval, Chapter 3*, volume 463. ACM press New York, 1999c.

R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern Information Retrieval, Chapter 2*, volume 463. ACM press New York, 1999d.

R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern Information Retrieval, Chapter 1*, volume 463. ACM press New York, 1999e.

Michael Buckland and Fredric Gey. The relationship between recall and precision. *Journal of the American society for information science*, 45: 12–19, 1994.

H. Chen and S. Dumais. Bringing order to the web: Automatically categorizing search results. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 145–152. ACM, 2000.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

Erik Hatcher, Otis Gospodnetic, and Michael McCandless. *Lucene In Action, Chapter 1.* Manning Publications, manning early access program edition, 2009.

W. Hersh, R.T. Bhuptiraju, L. Ross, P. Johnson, A.M. Cohen, and D.F. Kraemer. Trec 2004 genomics track overview. In *Proceedings of TREC*, volume 2004. Citeseer, 2004.

W.R. Hersh, R.T. Bhupatiraju, L. Ross, P. Roberts, A.M. Cohen, and D.F. Kraemer. Enhancing access to the bibliome: the trec 2004 genomics track. *Journal of Biomedical Discovery and Collaboration*, 1(1):3, 2006.

D.V. Johannsson. Biomedical information retrieval based on document-level term boosting. Master's thesis, Norwegian University of Science and Technology (NTNU), 2009.

K. Krishna, N. Murty, et al. Genetic k-means algorithm. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 29(3):433–439, 1999.

H.K. Le. Inductive clustering: A technique for clustering search results.

T.A. Louis. Finding the observed information matrix when using the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 44(2):226–233, 1982.

C.D. Manning, P. Raghavan, H. Schutze, and Ebooks Corporation. *Introduction to Information Retrieval, Chapter 16*, volume 1. Cambridge University Press Cambridge, UK, 2008a.

C.D. Manning, P. Raghavan, H. Schutze, and Ebooks Corporation. *Introduction to Information Retrieval, Chapter 8*, volume 1. Cambridge University Press Cambridge, UK, 2008b.

G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. LibreDigital, 2008.

C.L. Ngo and H.S. Nguyen. A method of web search result clustering based on rough sets. *IEEE Computer Society*, 2005.

Heri Ramampiaro. Biomedical information retrieval: The biotracer approach. *S. Khuri, L. Lhotska and N. Pisanti (Eds): ITBAM 2010, LNCS 6266*, pages 143–157, 2010.

A. Spink, D. Wolfram, M.B.J. Jansen, and T. Saracevic. Searching the web: The public and their queries. *Journal of the American Society for Information Science and Technology*, 52(3):226–234, 2001.

A. Spink, B.J. Jansen, D. Wolfram, and T. Saracevic. From e-sex to e-commerce: Web search changes. *Computer*, 35(3):107–109, 2002.

M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. 400:525–526, 2000.

H. Toda and R. Kataoka. A search result clustering method using informatively named entities. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 81–86. ACM, 2005.

H.J. Zeng, Q.C. He, Z. Chen, W.Y. Ma, and J. Ma. Learning to cluster web search results. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217. ACM, 2004.