Genomic Information Retrieval through Selective Extraction and Tagging by the ASU-BioAI Group

Lian Yu, Syed Toufeeq Ahmed, Graciela Gonzalez, Brandon Logsdon, Mutsumi Nakamura, Shawn Nikkila, Kalpesh Shah, Luis Tari, Ryan Wendt, Amanda Zeigler, Chitta Baral*
Department of Computer Science and Engineering,
Arizona State University,
PO Box 878809,
Tempe, AZ 85287-8809, USA

ABSTRACT

In this paper we describe the approach used by the Arizona State University BioAI group for the ad-hoc retrieval task of the TREC Genomics Track 2005. We pre-process TREC query expression by adding the synonyms of genes, diseases, bio-processes, functions of organs, and selectively adding stemming verbs, nouns, and Mesh Heading categories. The pre-processed queries are used to perform initial search on the TREC Genomics collection of MEDLINE abstracts and produce a set of target abstracts using Apache Lucene. Tagging, anaphor resolution and fact extraction are performed on the target abstracts to refine the search results in terms of relevance. Finally, we rank the target abstracts according to the extracted facts, distance between terms and terms appeared in the query.

1 INTRODUCTION

The BioAI Research Group of the Fulton School of Engineering in Arizona State University participated in the Ad-hoc Retrieval Task of the TREC (Text Retrieval Conference) Genomics Track in 2005. Provided were a set of retrieval queries collected from biologists that conformed to a set of generic topic templates (GTTs). There are 5 GTTs, each of which has 10 instances, for a total of 50 topics. Following is a list of the 5 GTTs listed as given, (available at http://ir.ohsu.edu/genomics/2005protocol.html) with the semantic types in each GTT underlined:

1. Find articles describing standard methods or protocols for doing some sort of experiment or procedure.
2. Find articles describing the role of a gene involved in a given disease.
3. Find articles describing the role of a gene in a specific biological process.
4. Find articles describing interactions (e.g., promote, suppress, inhibit, etc.) between two or more genes in the function of an organ or in a disease.
5. Find articles describing one or more mutations of a given gene and its biological impact.

The dataset for the TREC 2005 Genomics Track consists of completed citations from the MEDLINE database inclusive from 1994 to 2003. Records were extracted using the Date Completed (DCOM) field for all references in the range of 19940101 - 20031231. This provided a total of 4,591,008 records, which is about one third of the full MEDLINE database. The subset of articles provided by TREC is available in the "MEDLINE" format, which consists of ASCII text with fields indicated and delimited by different 2-4 character abbreviations.

The aim for our first participation in the TREC Genomics Track was to provide an efficient approach to retrieve most relevant abstracts from the subset regarding to the queries. We found that about 25% of MEDLINE records do not have an abstract, mainly because the article itself does not have one. We focused our retrieval task on the remaining 75%, giving us close to 3.5 million abstracts. A guiding principle for us was that relevance of a topic should not be just based on individual terms or keywords, such as genes or diseases, but rather it should take into account the subject of the whole document. In order to implement this principle, we would first parse the abstract to identify complete *facts*: the right semantic terms plus the right relationship among them, as specified in the query topic. We would extract those facts as a whole, noting that they might appear more than once in the abstract, and then take both fact and term frequency into consideration when ranking the abstracts for relevance.

The idea was that if we could extract all relevant facts from each abstract, we would just need to search among those extracted facts for those closely related to the query (rank) and return results. However, we needed to process a large volume of abstracts within a limited time in order to submit experiment results on time. We decided to retrieve a set of potential target abstracts from the given dataset using the given query topics, and then apply the extraction and ranking schema on that reduced set.

We choose Apache Lucene [1] to perform the initial retrieval. Apache Lucene is a high-performance, full-featured text search engine library written in Java. It is a technology suitable for an application that requires full-text search, especially cross-platform. Apache Lucene is an open source project available from Apache Jakarta. Figure 1 shows the architecture of our approach:

[*] To whom correspondence should be addressed.

1. Pre-processing queries: To apply Lucene in the biomedical domain, we needed to first incorporate bio-domain knowledge into the Lucene queries. For example, we needed to pre-process each of the 50 queries by adding synonyms, alias, and acronyms to genes, diseases, bio-processes, and functions of organs, as well Mesh categories information.
2. Indexing: uses Lucene indexing APIs to create a comprehensive index of terms on about 3.4 million of the given subset of MEDLINE abstracts.
3. Retrieval of target abstracts: uses a batch query-process Java program to input each pre-processed query topic and output a list of PubMed IDs (hereafter called PMIDs) associated to it. The size of target abstracts is narrowed down to 12K MEDLINE abstracts through this process.
4. Tagging: tags entities such as genes and diseases to facilitate anaphor resolution and fact extraction.
5. Resolving Anaphors: resolves pronouns of genes, diseases, and bio-processes so that the text extraction tool can extract facts of interest.
6. Fact extraction: extracts facts in terms of the relation identified from the queries.
7. Ranking of abstracts: we define a formula which takes the fact frequency, the distance of terms, as well as terms frequency into consideration.

In the following section, we expand on each step in this process. In Section 3 we describe the evaluation of our approach. In Section 4 we sketch future research directions

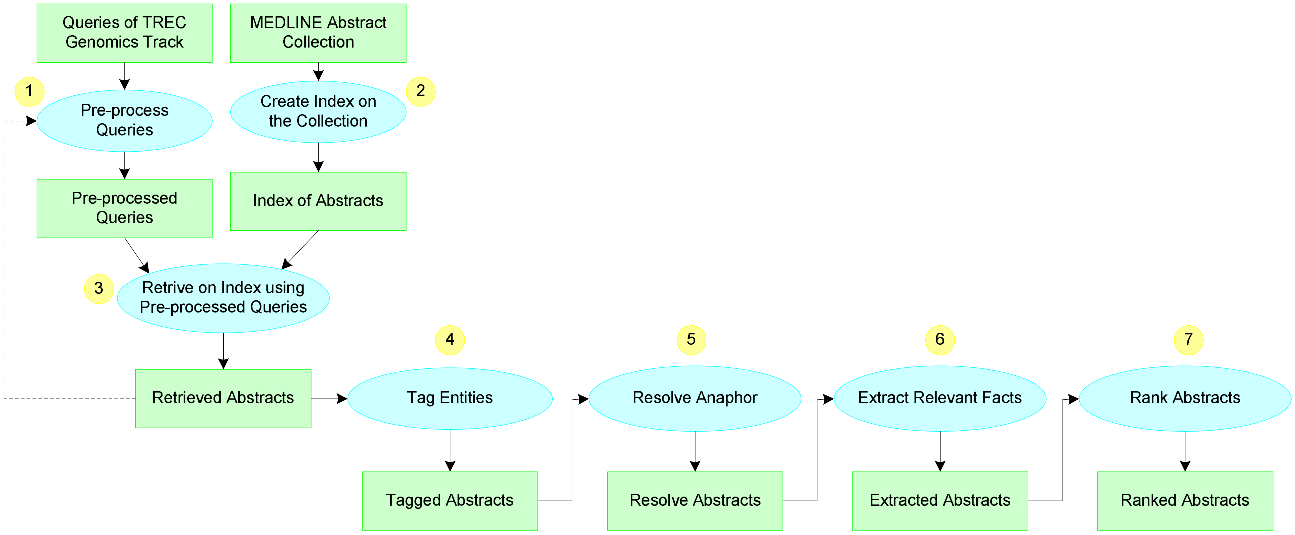


Fig. 1. Architecture of the Ad-hoc Retrieval System

2 INFORMATION RETRIEVAL SYSTEM

2.1 Pre-process Queries

Query pre-processing can be divided into three phases: synonym matching, stemming and fine tuning. Both the synonym matching and stemming phases are automatic, while the fine tuning phase is manually done based on the topics and the number of abstracts retrieved. We elaborate on each of them below. Since we use Lucene as our indexing system, the queries follow the Lucene syntax.

2.1.1 Synonym Matching

Given a list of words provided by TREC for each topic, synonym matching automatically checks if the given word is a gene or a disease. If it is either of the two, all of the corresponding synonyms are extracted from either Entrez Gene[1] or MeSH[2], respectively. Entrez Gene [6] is a gene database from NCBI, and MeSH [4] (short for Medical Subject Headings) is an ontology of terms used for categorizing articles in PubMed [6]. Both Entrez Gene and MeSH provide flat files available at their FTP sites. For topics that involve biological processes and functions, a selected set of terms from MeSH is used with their corresponding synonyms.

Synonyms are grouped by “OR” Boolean operator and groups of synonyms are connected with “AND” Boolean operator. For instance, suppose *g* is a gene name, and *g1* and *g2* are synonyms of *g*, and *d* is a disease name while *d1* is a synonym of *d*. Then the query formed would be:

```
(g OR g1 OR g2) AND (d OR d1)
```

[1] ftp://ftp.ncbi.nlm.nih.gov/gene/

[2] http://www.nlm.nih.gov/mesh/filelist.html

2.1.2 Stemming

Words that are not identified as genes or diseases are stemmed using the Porter Stemming algorithm [9], which returns the root form of a word. A wildcard is attached to each stemmed word to form the query. For instance, the word “progression” is stemmed as “progress” and “progress*” is formed as part of the query.

2.1.3 Fine-Tuning

Queries formed by the previous two phases can result in a large number of relevant abstracts for some of the topics. Furthermore, our queries have to reflect the specific needs of TREC. Therefore, it is insufficient to have the keywords and their synonyms as part of the queries. It is necessary to add extra information to the queries. For example, in the case of topic 110, we are interested in the role of interferon-beta gene in Multiple Sclerosis. Using “interferon-beta”, “multiple sclerosis” and their corresponding synonyms as a query would retrieve articles that used interferon-beta as a treatment, which is not of our interest.

The fine tuning approach differs in the various templates. In template 1 about methods and protocols, if the given keywords appear in MeSH, the query is modified so that the keywords must appear as the MeSH heading of the abstracts. MeSH headings act as categories of the articles. For instance, in topic 100, electroporation is a MeSH heading, so the query for topic 100 contains “MH – electroporation” to make use of the MedLine format of the abstracts. The query for topic 100 is formed as follows in Lucene syntax:

```
    "MH \- Electroporation" AND "cell" AND
"open"
```

For template 2 regarding the role of genes in diseases, MeSH headings such as genetics and pathology are added in the queries. In the case of topic 118 regarding the gene TGFB and Cerebral Amyloid Angiopathy, we added “genetics AND pathology” as part of the query. The query is formed as follows:

```
  (TGFB1 OR CED OR DPD1 OR TGFB OR beta 1)
AND ("Cerebral Amyloid Angiopathy" OR
"Congophilic Angiopathy" OR "Sporadic
Cerebral Amyloid Angiopathy" OR "Cerebral
Amyloid Angiopathies" OR "Congophilic
Angiopathies") AND pathology AND genetics
```

On the other hand, NOT operators ‘-’ are used on words such as treatment and clinical trials as part of the queries in template 2 to exclude them from the search. The query for topic 110 regarding interferon-beta and multiple sclerosis is formed as follows:

```
  ("interferon* beta*") AND ("Multiple
Sclerosis" OR "MS" OR "Disseminated
Sclerosis") AND -"PT - Clinical Trial" AND
-treat* AND -therap*
```

Similarly, words such as polymorphism and mutation are added as part of the queries for template 5, to reflect the fact that we are interested in only articles about mutation of genes. The query for topic 143 regarding the mutation of NM23 and tracheal development is formed as follows:

```
  ("NME1" OR "AWD" OR "GAAD" OR "NDPKA" OR
"NM23" OR "NM23\-H1") AND (("tracheal
development" OR "tracheal develop*")) AND
(polymorphism OR mutation)
```

2.2 Indexing

Using the Standard Analyzer provided by Lucene, it tokenizes the words in the abstracts to perform indexing. The process of tokenization involves the use of stop-word list, so that frequently used but uninformative words, such as a, an, the, would not be used for indexing. The Lucene index stores the tokens and a list of files in which each of the token appears.

2.3 Entity Tagging

The task is to parse a biomedical text to identify entities such as diseases, biological processes and biological functions, and then tag them accordingly with DISE_<term>, PROC_<term>, and FUNC_<term>, where <term> represents the name of the disease, biological process, or biological function, usually consisting of several words. Abner [11], a system based on statistical machine learning techniques, was used to identify gene and protein names.

Two problems arise when dealing with this task. The first problem is the looseness of the English language in conjunction with synonyms, alias, acronyms and even spelling errors. Tagging can not be done by simply matching token by token because of names spanning multiple words, so we must consider one or multiple words when making comparisons. The second problem is efficiency. Brute force methods will allow us to tag all instances of biological words; however, the running time makes it an unrealistic choice for the amount of abstracts with which we have to deal.

We tried to exclude as many words as possible from being matched such that the only words that are matched are words of meaning and content. In the English language, nouns, verbs, adjectives, and adverbs convey the real meaning and content. Everything else just works to make a more readable sentence, and they never change the content.

In order to reduce the number of word groups matched we chunked words into noun and verb groups. Recognizing that most likely an entity of interest would be either a noun phrase (adjective/noun) or a verb phrase (adverb/verb) we grouped words using a part-of-speech tagger tool called Monty Tagger [5] into sequences. Thus, given an abstract, before attempting to tag biological terms, we tag parts of speech using the Monty Tagger Java API. An abstract where each of the words is tagged with a part of speech preceded by '/' looks as follows:

```
Haemopoietic/NNP cell/NN growth/NN and/CC
differentiation/NN is/VBZ primarily/RB
regulated/VBN by/IN the/DT local/JJ
production/NN of/IN various/JJ
cytokines/NN within/IN the/DT bone/NN
marrow/NN micro_environment/NN 3/CD ,/,
as/IN well/RB as/IN by/IN the/DT
circulating/VBG hormone/NN ,/,
erythropoietin/NNP GENE_EPO/NNP ./.
```

This tagged string is then tokenized and one by one analyzed for its part of speech in order to be grouped. There are four main parts of speech that we care about: Noun, Verb, Adjective, and Adverb. Nouns are tagged with /NN, /NNS, /NNP, and /NNPS. Verbs are tagged with /VB, /VBD, /VBG, /VBN, /VBP, and /VBZ. Adjectives are tagged with /JJ, /JJR, and /JJS. Adverbs are tagged with /RB, /RBR and /RBS. In the example above, the first noun group is, "Haemopoetic/NNP cell/NN growth/NN". This group is then matched against our dictionaries of diseases, biological processes, and functions (compiled from MeSH). If there is a direct match the string is tagged as a Disease, Process, or Function respectively. The same process is applied to verb groups. In the above example the first verb group is "is/VBZ primarily/RB regulated/VBN". The tagged abstracts are then used for anaphor resolution.

2.4 Anaphora Resolution

In linguistics, an anaphora is an expression that is used to refer back to some entity (or entities). Pronouns (such as it, their, this) are the most common anaphora, though other pro-forms are also anaphoras. The entity to which an anaphora refers is its *referent* or *antecedent*. Consider:

"Luis sent me an email. It was the first thing he did that day."

Here, both "it" and "he" are anaphoras that refer back to the email and Luis (the antecedents), which were mentioned before. A human reader has no problem identifying what they refer to, but automatic processing of the text requires their resolution: that is, finding a potential replacement for them and substituting the anaphoras. For the biomedical domain, we also need to apply some semantic information to accurately replace some anaphoras.

The subtasks for anaphor resolution include creating a referent candidate list, doing a proximity search, and finding the longest common substring to identify the right antecedent. We elaborate on each of them next.

2.4.1 Creating a referent candidate list

A list of referents is maintained as they are encountered in the sequential parsing of the abstract. The number of referents can grow very large and prohibit efficient and sensible search for resolution of ambiguous anaphora. So the list is limited to only those words that are tagged as potential names for genes, proteins or diseases. To facilitate this, a variable is kept to track the most recent reference to an antecedent entity, since sometimes there are some anaphoras that are used more than once (like using "he" or "it" in subsequent sentences in the example above).

Semantic Chunk Objects (SCO) [2] are the potential candidates for the anaphoras, and contain a potential antecedent plus information such as the distance of the SCO from the first word in the abstract, the score received by the SCO as a potential candidate for the anaphora and semantic information such as whether the SCO is a gene, disease or protein.

2.4.2 Proximity Search

Information regarding sentence number and distance from the beginning of the text is kept for each SCO. Usually the correct antecedent is the closest one to the anaphora. Using the scoring heuristic and semantic information along with the proximity information helps better resolve the anaphoras.

2.4.3 Longest Common Substring

Anaphoras such as "these" or 'both" are resolved by looking at the word that follow them. A longest common substring comparison is run on that word and the potential SCOs. Depending upon how long the common substring is one can decide which semantic chuck object the anaphora substitutes. Consider the example:

"The exon1 and exon3 are most crucial in expression of these genes. These exons are…".

In this case the word next to "these" is exon and that is compared to all the SCO's finding the noun group exon1 and exon3 as the closest matches thus are replacement for anaphora "these".

The comparison is run not only on the antecedent attribute of the SCO but also on the semantic information of the SCO. Consider for example:

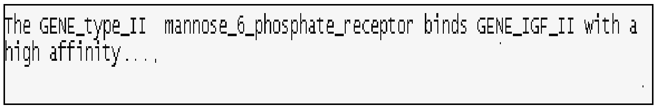
"Alzheimer's disease and variant Creutzfeldt-Jakob disease affect the brain. These diseases are more prominent…"

The comparison will include the word following "these" i.e. "disease" and thus we get the group "Alzheimer's disease and variant Creutzfeldt-Jakob disease" as the potential replacement.

2.5 Fact Extraction

As explained before, we consider a "fact" the entity or set of entities participating in a relationship of interest for the topic. Once the entities are recognized through tagging –Section 2.3- and anaphor resolution –Section 2.4-, the abstracts are ready for fact extraction. Recall that the abstracts are tagged with the following information (an example is shown in Fig. 2):

- Gene names as GENE_<phrase of a gene name>
- Diseases as DISE_< phrase of a disease>
- Biological Processes as PROC_< phrase of a biological process >
- Molecular Functions as FUNC_< phrase of a molecular function>



The GENE_type_II mannose_6_phosphate_receptor binds GENE_IGF_II with a high affinity....

Fig. 2: An example of tagging

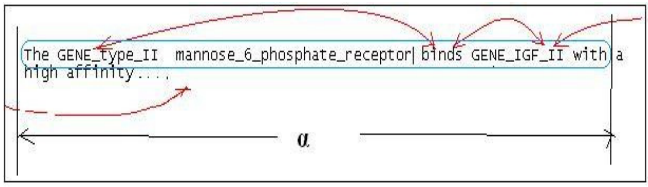
Extraction of facts corresponding to the templates (shown below) is carried out by the extraction module, where the idea of lexical chaining is applied. The premise is that words that are closer to each other are more likely related than the ones that are far apart. The general schema for extracted facts is as follows:

PMID| Fact Id| Fact Frequency| GENE_< phrase >| Interaction Word | GENE_< phrase > | DISE_< phrase >| PROC_< phrase > | FUNC_< phrase > |

where PMID is a unique PMID, Fact Id is unique fact number within the given abstract, Fact Frequency is frequency of the fact occurring in the given abstract, GENE_< phrase > is gene name prefixed with GENE_, similarly DISE_< phrase >, PROC_< phrase > and FUNC_< phrase > is for disease, process and function respectively.

Extraction Solution

A lexical chain [8] is a lexical cohesion of related words that contribute to the continuity of meaning. Based on this idea, the extraction module tries heuristically to construct a chain of related words (such as gene names and diseases) and include them as a fact only if they are considered to be related. The relation of these words is primarily assumed to be in a single sentence.



The GENE_type_II mannose_6_phosphate_receptor| binds GENE_IGF_II with a high affinity....

Fig. 3: Heuristic based on Lexical (semantic) cohesion of words. Windowing technique is used to limit the chain size under consideration.

The method can be explained with the help of windowing technique. Let's consider the input text (sentence by sentence) to be a single stream of text with a window of size α. A chain of related terms that fit within this window size is a potential candidate for fact extraction. The extraction module now considers the terms lexically related within this window. This window is moved one word at a time throughout the text. Fig. 3 shows the window and the input text. The fact extraction task is performed on all gene-related TREC topics (i.e. all topics except topic 1).

2.6 Ranking Abstracts

The Ranking module takes two input files: 1) a query file which contains a TREC template or query ID and corresponding queries (created earlier in the Query Pre-processing step) and 2) a query results file, which contains a list of query IDs and associated PMIDs returned from a Lucene search. For each query ID, the ranking module outputs a ranking of the PMIDs based on relevancy.

To rank the abstracts with for a given query ID, a hierarchical ranking approach is used in this paper: the abstracts associated with a query ID are first ranked based on the number of times relevant facts occur in each abstract. This measure is called the fact frequency. The fact frequency counts the number of times that related terms in the query appear in a same sentence (see description of the Extraction module) in the abstracts of interest. If any two abstracts have the same fact frequency the tie is broken by comparing the abstracts' term/distance scores, abbreviated TD. (see Equation 2). The TD score incorporates both the term frequency and term distance (where terms are not necessarily in the same sentence, unlike fact frequency) of the abstracts in order to determine which abstract is more relevant.

TREC requires that each abstract be assigned a single score to represent its relevancy. To accomplish this, a ranking formula called *rf(a)* is used to assign a value to an abstract *a* that meets the above two requirements.

If *A* represents all the abstracts that have the same query ID, $a \in A$, *ff(a)* represents the fact frequency of *a*, and *td(a)* represents the TD score of *a*, then the value *rf*(*a*) is as follows in Equation (1):

$$rf(a) = td(a) + \lceil \max(td(A)) \rceil * ff(a) \quad (1)$$

In Equation 1 the maximum score of all the abstracts in *A* is used in order to ensure that an abstract that has a high fact frequency, and is thus more relevant, will have a high value of *rf(a)*. The TD score of the abstract is also taken into account to break ties between abstracts with identical fact frequencies, resulting in a single value which ranks an abstract using the hierarchical ranking approach described earlier.

Once *rf(a)* for each abstract has been computed, the list of abstracts, sorted by query ID and then sub sorted in decreasing order by *rf*(*a*), are written to two files, qrels.txt and topic_document.txt, which were then submitted to TREC for evaluation.

The notations used in the following formula are listed as follows:

a: an abstract with a query ID of *ID*

A: the set of all the abstracts with a query ID of *ID* and $a \in A$.

G: the set of all genes and their synonyms found in the query *ID*

D: the set of all the diseases and their synonyms found in the query *ID*

s(*a*): the number of sentences in *a*

$d = (s_1, s_2)$: the distance between strings s_1 and s_2 (such as strings of a gene or a disease) in terms of sentences in between

Min_a (*d*): the minimum distance in sentences between s_1 and s_2 If either s_1 or s_2 is not in *a* then Min_a (*d*) is defined to be equal to s(*a*)

f(*a*, *s*): the number of times the string *s* occurs in *a*

w(*a*): the total number of words in *a*

df(*a*, *d*): the number of times the pair of strings of *d* is at a distance of Min_a (*d*) (if either string in *d* is not in *a* then df(*a*, *d*) is defined to be equal to 1).

Then the TD score of *a*, *td(a)*, is computed as follows in Equation (2).

$$td(a) = \begin{cases} ratio * dist * rel, & if \quad w(a) > 0 \wedge s(a) > 0 \\ 0, & if \quad w(a) = 0 \vee s(a) = 0 \end{cases}$$
(2)

where the variable *ratio* is computed with gene or disease overlap and the weights of the genes or diseases in the query *ID*, *dist* represents the distance between gene/disease pairs in the query *ID* (see Equation (3)). *rel*, or relevancy, represents the frequency of the genes and diseases in the query *ID* appearing in *a* (see Equation (6)).

The portion of ratio in TD score consists of the product of three factors as follows in Equation (3)[1]:

$$ratio = \frac{overlap}{|G| + |D|} * \frac{1}{\sqrt{w(a)}} * weight \qquad (3)$$

The first factor is called the coordination, which is the *overlap* (see Equation (4)) divided by the number of genes and diseases in the query *ID*. The second factor is known as length normalization, which is the reciprocal of the square root of the total number of items, tokens, or the words of the abstract searched. The last factor, weight, represents the weights assigned to each gene and disease in the query *ID*: since each gene and disease has equal weight this score will be the square root of the number of genes and diseases in the query *ID* (see Equation (5)) [1].

The *overlap* of an abstract *a* is the number of genes or diseases in the query *ID* that occur at least once in *a*.

$$overlap = \left| d \in G \cup D : f(a,d) > 0 \right| \qquad (4)$$

The *weight* value normalizes the weights for the genes and diseases in the query *ID*.

$$weight = \frac{1}{\sqrt{\sum_{d \in G \cup D} wgh(d)^2}} \qquad (5)$$

where w*gh*(*d*) represents the individual weight of a gene or disease *d* in the query *ID*: by default, this value is equal to one divided by the number of genes and diseases in the query *ID*.

The distance, *dist*, is the product of the distance scores for each possible ordered pair of genes and diseases in the query *ID.* A distance score for a gene/disease pair is a product of two factors: a sentence distance factor and a distance frequency factor. The sentence distance factor is a value between one and two inclusive, with a value of one signifying that the gene/disease pair do not occur together in the abstract *a* and a score of two signifying that the gene and disease occur in the same sentence in the abstract *a*. The value of the distance frequency factor is how often the gene/disease pair occurs at the distance used to calculate the sentence distance factor: the square root is applied to this value for normalization. If *ID* is between a certain range the query includes not one but two sets of genes (G_1 and G_2) for which the distance must be taken into account (see corresponding template): therefore, a second product term must be included which measures the distance between each possible ordered pair of genes in these two sets.

The distance (*dist*) value is the product of the distance values between each possible ordered pair of genes and diseases in the query *ID.* A distance value for a gene/disease pair is itself a product of two factors: a sentence distance factor and a distance frequency factor. The sentence distance factor is a value between one and two inclusive: one signifying that the gene/disease pair does not occur together in the abstract *a* and two signifying that the gene and disease occur in the same sentence in the abstract *a*. The value of the distance frequency factor indicates how often the gene/disease pair occurs at the distance used to calculate the sentence distance factor, and the square root is applied to this value for normalization. If *ID* is between 130 and 139 then Equation (6b) is used to calculate the value of *dist*. Within this range the query includes not one but two sets of genes (G_1 and G_2) for which the distance must be taken into account (see corresponding template): therefore, along with the product found in Equation (6a), a second product term must be included which measures the distance between each possible ordered pair of genes in these two sets.

$$dist = \prod_{d \in G \times D} (2 - \frac{d(a,d)}{s(a)}) * \sqrt{df(a,d)}$$

$$(T < 130 \vee T > 139) \wedge s(a) \neq 0 \quad \text{(6a)}$$

$$dist = \prod_{d \in G \times D} (2 - \frac{d(a,d)}{s(a)}) * \sqrt{df(a,d)} *$$

$$\prod_{g \in G1 \times G2} (2 - \frac{d(a,g)}{s(a)}) * \sqrt{df(a,g)}$$

$$T \geq 130 \wedge T \leq 139 \wedge s(a) \neq 0 \quad \text{(6b)}$$

The relevancy portion in TD of *a*, *rel*, is a sum of the relevancy of each gene or disease searched for in *a*. The relevancy of an individual gene or disease is a product of its frequency and its inverse document frequency [3],[10]. The square root of the frequency is taken to normalize the value, while the inverse document frequency measures how rare a gene or disease is: the rarer the gene or disease, the higher the inverse document frequency, and thus the relevancy, will be. This is because the occurrence of a rare gene or disease in an abstract is a better indicator of relevancy than a common one.

$$rel = \sum_{d \in G \cup D} \sqrt{f(a,d)} * \log(\frac{A}{dfr(A,d) - 1}) \quad (7)$$

The document frequency, *dfr*(A,d) (see Equation (8)), represents the number of abstracts in *A* that contain the gene or disease *d* at least once. The smaller the value of *dfr*(*A*,*d*) the rarer the gene or disease *d* is.

$$dfr(A,d) = \left| a \in A : f(a,d) > 0 \right| \quad (8)$$

3 EVALUATION AND DICUSSION

This section demonstrates the experiment results of our approach, and the comparison with results from Pubmed search engine in regarding to the 50 topics.

3.1 Run Results

We performed Lucene indexing on the collection of 4.5 million abstracts in MEDLINE format, used 50 formatted queries to search in the Lucene index, and retrieved about 12K target abstracts. The number of search results varied from query to query ranging from 0 to 7,000.

For each template, we performed tagging, anaphor resolution, extraction (exception template 1) and ranking. We performed fact extraction on abstracts pertinent to templates 2 through 5, and calculated the fact frequencies, which were incorporated into the ranking formula. For abstracts related to template 1, we performed tagging and anaphor resolution without fact extraction. The ranking is only dependent on the score as described in Section 2.5.

3.2 TREC Evaluation Results

TREC ad-hoc relevance judgments were done based on the top 60 documents from the two runs submitted by each group for each topic, which yielded an average pool size of 822 documents. Relevance judgments were performed only on 49 of the 50 topics, as no relevant document was found for one of the topics, which is topic #135. Evaluation results returned by TREC were summarized in terms of precision at top 10 relevant documents retrieved (denoted as P10), precision at top 100 relevant documents retrieved (P100) and uninterpolated average precision for the 49 topics.

Our system for the ad hoc retrieval task achieves an overall precision of 0.2714 for P10 and precision of 0.1061 for P100 among the 49 topics. This implies that our system achieves a low recall, as the number of articles retrieved by our system is low. Table 1 shows the average number of articles retrieved as well as minimum and maximum number for each template. We further analyzed our performance and noticed that our system performs best in template 2, which is to retrieve articles describing the role(s) of a gene involved in a disease. It is evident that our extraction-based retrieval system benefits from the rich dictionaries of gene and disease names compiled from Entrez Gene and MeSH. On the contrary, the lack of rich dictionaries for functions of an organ (for template 4) and biological impact or role (for template 5) is the main reason on why our system suffers in the precision of the retrieval task for templates 4 and 5. Our system also failed to retrieve any documents for some of the topics in templates 4 and 5.

Table 1: Average of the number of articles retrieved

Template #	Number of articles retrieved on average
1 (topic # 100-109)	95 (min = 2, max = 436)
2 (topic # 110-119)	110 (min = 4, max = 303)
3 (topic # 120-129)	122.5 (min = 4, max = 1000)
4 (topic # 130-139)	8.11 (min = 0, max = 52)
5 (topic # 140-149)	26.9 (min = 0, max = 207)

4 CONCLUSION AND FUTURE WORK

The TREC Genomics team included 8 members from the BioAI group, 3 of them undergraduates. We spent about 6 weeks completing the ad-hoc retrieval task. We used Apache Lucene to perform the initial retrieval and got 12K target abstracts out of 4.5 million abstracts. Then tagging, anaphor resolution, extraction and ranking were performed to refine relevance of search results.

Retrieval systems such as NCBI PubMed generally return a large number of documents that are supposed to be relevant to the users' queries. In other words, such retrieval systems achieve high recall but relatively low precision. Users of a retrieval system with high precision can benefit on the preciseness and conciseness of the articles returned to them. With this in mind, we emphasized the precision aspect of our retrieval system. Our system could achieve higher precision on templates 4 and 5 if richer ontologies were used for functions of an organ and biological impact.

Future work includes investigating ways to improve the accuracy of the tagging module for diseases, bio-processes and functions of organs. A quantitative approach to assign scores to SCOs is needed for the anaphor resolution module. The extraction module needs to be able to extract various types of the topics, such as topics in the template 1.

ACKNOWLEDGEMENTS

REFERENCES

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

[2] J. Castaño, J. Zhang, J. Pustejovsky. Anaphora Resolution in Biomedical Literature. *International Symposium on Reference Resolution*, 2002.

[3] K.S. Jones. Index term weighting. *Information Storage and Retrieval*, **9**: 619-633, 1973.

[4] MeSH. http://www.nlm.nih.gov/mesh/.

[5] Monty Tagger. http://web.media.mit.edu/~hugo/montytagger/.

[6] NCBI Entrez Gene. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene.

[7] NCBI Entrez PubMed. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=PubMed.

[8] J. Morris, G. Hirst, Lexical Cohesion Computed by Thesaural Relations as an indicator of the structure of Text. *Association of Computational Linguists*, 1991.

[9] M.F. Porter. An algorithm for suffix stripping, *Program*, **14**(3):130-137, 1980.

[10] S.E. Robertson and K.S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, **27**, 129—146, 1976.

[11] B. Settles. ABNER: an open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, **21**(14):3191-3192, 2005.