Domain-Specific Synonym Expansion and Validation for Biomedical Information Retrieval (MultiText Experiments for TREC 2004)

Stefan Büttcher Charles L. A. Clarke Gordon V. Cormack

School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

Abstract

In the domain of biomedical publications, synonyms and homonyms are omnipresent and pose a great challenge for document retrieval systems. For this year's TREC Genomics Ad hoc Retrieval Task, we mainly addressed the problem of dealing with synonyms. We examined the impact of domain-specific knowledge on the effectiveness of query expansion and analyzed the quality of Google as a source of query expansion terms based on pseudo-relevance feedback.

Our results show that automatic acronym expansion, realized by querying the AcroMed database of biomedical acronyms, almost always improves the performance of our document retrieval system. Google, on the other hand, produced results that were worse than the other, corpus-based feedback techniques we used as well in our experiments. We believe that the primary reason for Google's bad performance in this task is its highly restricted query language.

1 Introduction

This paper describes the work done by members of the MultiText project at the University of Waterloo for the TREC 2004 Genomics track. The MultiText group only participated in the Ad hoc retrieval task of the Genomics track, which consisted of finding documents that are relevant with respect to certain topics in a subset of the Medline/PubMed database of medical publications [PM04]. The subset included articles published between 1963 and 2004. It consisted of 4,591,008 different documents with a total size of 14 GB (XML text), where the term *document* refers to incomplete articles, annotated with certain keywords, such as MeSH terms [MeS04]. Out of these 4,591,008 documents, 3,479,798 contained the abstract of the article, the remaining 1,111,210 only the article title and the annotations. In total, 50 topics were given to the track participants. An example topic is shown in Figure 1.

Information retrieval in the context of biomedical databases traditionally has to contend with three major problems: the frequent use of (possibly non-standardized) acronyms, the presence of homonyms (the same word referring to two or more different entitities) and synonyms (two or more words referring to the same entity). For the two runs we submitted for the Genomics track, we addressed the problems caused by acronyms and synonyms. We chose to follow a strategy that was a mixture of known heuristics for approximate string matching for genomics-related data and a knowledge-based approach that used domain-specific knowledge from various sources, such as the AcroMed database of biomedical acronyms [Acr04]. Since acronyms can be viewed as special forms of synonyms, it was possible to develop a system that can deal with both problems (synonyms and acronyms) in a uniform way.

Furthermore, we implemented an automatic query expansion technique using pseudo-relevance feedback. We evaluated the quality of feedback terms produced by a simple Google query and compared the results with known query expansion techniques. We also studied possibilities to combine both strategies.

```
<TOPIC>
  <ID>14</ID>
  <TITLE>Expression or Regulation of TGFB in HNSCC cancers</TITLE>
  <NEED>
    Documents regarding TGFB expression or regulation in HNSCC cancers.
  </NEED>
  <CONTEXT>
    The laboratory wants to identify components of the TGFB signaling
    pathway in HNSCC, and determine new targets to study HNSCC.
  </CONTEXT>
</TOPIC>
```

Figure 1: Example topic in original XML form

2 The MultiText system

Our work is based on the MultiText text retrieval system, which was initially implemented at the University of Waterloo in 1994 [CCB94] and has experienced a variety of extensions and modification over the last decade. MultiText supports several different document scoring mechanisms. We employed the MultiText implementation of Okapi BM25 [RWJ+94] [RWB98] and MultiText's QAP passage scoring algorithm [CCT00] [CCL01], which had initially been developed for question answering tasks.

2.1 Okapi BM25 document scoring

MultiText's implementation of BM25 is the same as the original function proposed by Robertson et al. [RWJ+94] in the absence of document relevance information. Every query term T is assigned a term weight

$$w_T = \ln(\frac{|\mathcal{D}| - |\mathcal{D}_T| + 0.5}{|\mathcal{D}_T| + 0.5}), \quad (1)$$

where $\mathcal{D}$ is the set of all documents in the corpus, and $\mathcal{D}_T$ is the set of documents containing the term T. For a given query $\mathcal{Q} = \{T_1, ..., T_n\}$, the score of a document D is computed using the formula

$$\sum_{T \in \mathcal{Q}} w_T \cdot q_T \cdot \frac{d_T \cdot (1 + k_1)}{d_T + k_1 \cdot ((1 - b) + b \cdot \frac{len_D}{len_{avg}})}, \quad (2)$$

where d_T is the number of occurrences of the term T in the document D, len_D is the length of D, len_{avg} is the average document length in the corpus, and q_T is the query-specific relative weight of the term T. Usually, q_T equals the number of occurrences of T in the original query. We discuss this parameter in more detail in section 8. The remaining parameters in formula 2 were chosen to be $k_1 = 1.2$ and $b = 0.75$.

In addition to mere term queries, MultiText supports structured queries, and in particular disjunctions of terms. We call a disjunction of query terms

$$T_\vee = T_1 \vee T_2 \vee \cdots \vee T_m$$

a *disjunctive query element*. The BM25 weight of a disjunctive query element is

$$w_{T_\vee} = \ln(\frac{|\mathcal{D}| - |\mathcal{D}_{T_1} \cup \cdots \cup \mathcal{D}_{T_m}| + 0.5}{|\mathcal{D}_{T_1} \cup \cdots \cup \mathcal{D}_{T_m}| + 0.5}). \quad (3)$$

For convenience, we let the "+" symbol denote the disjunction operator ("$\vee$") whenever we present example queries.

2.2 MultiText QAP passage scoring

MultiText's QAP passage scoring algorithm computes text passage scores based on the concept of self-information. It may be extended to a document scoring algorithm by finding the passage with the highest score within a document and assigning the passage's score to the entire document, although this is not the original purpose of QAP. Our pseudo-relevance feedback mechanisms are derived from the QAP scoring function.

Given a query $\mathcal{Q}$, every passage P of length l, containing the terms $\mathcal{T} \subseteq \mathcal{Q}$, is assigned the score

$$H(P) = \sum_{T \in \mathcal{T}} (\log_2(\frac{N}{f_T}) - \log_2(l)), \quad (4)$$

where N is the total number of tokens in the corpus (corpus size), and f_T is the number of occurrences of the term T in the corpus. $H(P)$ is called the *self-information* of the passage P.

3 System overview

An overview of the document retrieval system developed for the Genomics track is given by Figure

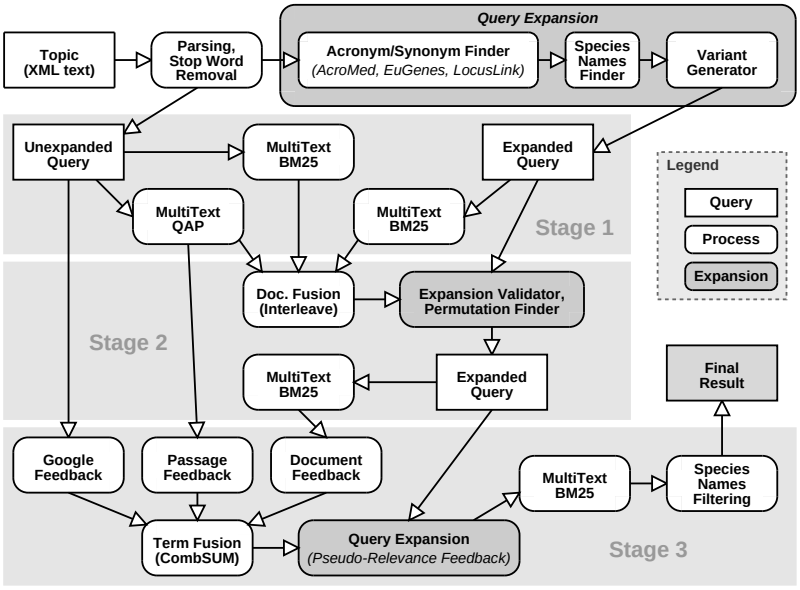

Figure 2: The document retrieval system

2. The system consists of a preprocessing stage and three document retrieval stages. Most parts of the system can be changed by varying the respective parameters. Because there were no training data available, many of these parameters are arbitrarily chosen.

Query preprocessing & Synonym expansion (Stage 1)

In the preprocessing stage, the input data (XML text) is parsed and all stop words are removed. The list of stop words contains the usual terms, like "the" and "and", but also a number of additional terms inferred from the sample topics, e.g. "find", "information", and "literature". After the stop words have been removed, several synonym generation techniques are applied to the query. A detailed explanation of these techniques can be found in section 4. The result of this process is an expanded query containing possible synonyms of the terms in the original query. The unexpanded query is taken by MultiText to score documents using both BM25 and QAP, while the expanded query is used to produce a third, BM25-scored document set.

Expansion validation (Stage 2)

In the second stage (see section 5 for details), the documents returned by the first stage are analyzed and systematically scanned for any occurrences of terms added to the query in the expansion phase of stage 1 or permutations of query terms. Terms that have been added to the query in stage 1, but occur very infrequently in the documents returned, are removed from the query, whereas frequent permutations of query terms which have not been part of the old query might be added. The resulting query is sent to MultiText, and documents are scored by BM25.

Pseudo-relevance feedback (Stage 3)

The third stage, which is described in sections 6 and 7, performs a set of standard pseudo-relevance feedback operations in order to create new expansion terms. The terms produced by the feedback are added to the query, and BM25 is used again to compute the document scores. Finally, if our retrieval system has found species names in the original query, it tries to match the documents in the result list against these species names and promotes documents that contain the species names or synonyms of them.

4 Domain-specific query expansion

For domain-specific query expansion, we used two different techniques in stage 1 of our system:

- a general variant generation approach that creates lexical variants of a list of given terms;
- a database of biomedical synonyms, extracted

```
ID: 14
TITLE: Expression or Regulation of TGFB in HNSCC cancers
NEED: Documents regarding TGFB expression or regulation in HNSCC cancers.
QUERY_TITLE: "expression" "regulation" "tgfb" "hnscc" "cancers"
EXPANDED_TITLE: "expression" "regulation" ("tgfb"+"blk"+"ced"+"dlhc"+
    "tgfbeta"+"tgfb1"+"tgfb2"+"tgf r"+"transforming growth factor beta"+
    "latent transforming growth factor beta"+"ltbp 1"+"ltbp1"+"mav"+
    "maverick"+.....) ("hnscc"+"head and neck squamous cell") "cancers"
QUERY_NEED: "regarding" "tgfb" "expression" "regulation" "hnscc" "cancers"
EXPANDED_NEED: "regarding" ("tgfb"+"blk"+"ced"+"dlhc"+"tgfbeta"+"tgfb1"+
    .....) "expression" "regulation" ("hnscc"+"head and neck squamous cell")
    "cancers"
```

Figure 3: Example topic after preprocessing and query expansion in stage 1

from various sources available on the web.

In addition to these methods, we examine permutations of query terms to find further expansions. The result of the query expansion in stage 1 is an expanded query that combines each original query terms and all its expansions (synonyms etc.) into one *disjunctive query element* (defined in section 2), as shown in Figure 3.

4.1 Lexical variants

In the context of biomedical articles, we have to deal with an abundant number of lexical variants of the same term. These are mainly caused by the authors' different preferences of hyphenation, spacing and Greek letters. In the Medline corpus used for this year's Genomics track, for instance, the NF-kappa B protein is referred to in 6 different ways: "NF-kappa B" (33902 times), "NF-kappaB" (28551), "NFkappaB" (3211), "NF-kB" (688), "NFkB" (259), and "NFkappa B" (45).

Our system addresses this problem by scanning the original query for interesting terms (i.e., terms that are likely to have lexical variants). We consider a term interesting if it contains at least one hyphen, digit, or Greek letter. For each such term, a list of possible variants is created according to the following rules:

1. Greek letters are contracted to their Latin equivalents: "alpha" becomes "a", "beta" becomes "b", and so on.
2. Hyphens are removed from the term.
3. Before and after every Greek letter, a hyphen is inserted.
4. At every transition from alphabetic to numerical characters and back, a hyphen is inserted.

For each term, every possible combination of the above rules is applied to the term. For example, the set of variants generated for the term "Lsp1alpha" (Larval serum protein 1 alpha) is: "lsp-1-alpha", "lsp-1-a", "lsp-1alpha", "lsp-1a", "lsp1-alpha", "lsp1-a", "lsp1alpha", "lsp1a".

The problem of replacing spaces by hyphens and vice versa can be ignored because the MultiText engine already considers them equivalent. Case folding is automatically applied to all terms as well.

4.2 Synonym expansion

In biomedical publications, we have mainly three types of synonyms that can appear in a query:

- acronyms;
- gene names or symbols;
- protein names or symbols.

Acronyms can be dealt with in a straightforward fashion by adding the respective long form to the query whenever a known acronym is encountered. Genes and proteins have additional structure that can be used when looking for appropriate expansions: the symbol "ACP" (acid phosphatase) describes a family of proteins, including ACP1 and ACP2. This allows us to not only add synonyms of "ACP" to the query, but also "ACP1" and "ACP2". Following the Gene Ontology terminology, we call these *narrow synonyms* (as opposed to *exact synonyms*, such as acronyms).

We used three different sources to deal with the above kinds of synonyms: The AcroMed database of biomedical acronyms [PCC$^+$01] [Acr04], the Eukaryotic Genes database at the University of Indiana [Gil02] [euG04], and the LocusLink database [LL04].

AcroMed

The AcroMed database contains a list of mappings from acronyms to their long forms, automatically

```
ID: 14
TITLE: Expression or Regulation of TGFB in HNSCC cancers
NEED: Documents regarding TGFB expression or regulation in HNSCC cancers.
QUERY_TITLE: "expression" "regulation" "tgfb" "hnscc" "cancers"
EXPANDED_TITLE: #1.00 "expression" #1.00 "regulation" #0.45 "tgfb" #0.95
    ("tgfb"+"tgf beta"+"beta tgf"+"tgfbeta"+"tgfb1"+"tgfb2"+"tgfb3") #0.45
    "hnscc" #0.95 ("hnscc"+"head and neck squamous cell") #1.00 "cancers"
QUERY_NEED: "regarding" "tgfb" "expression" "regulation" "hnscc" "cancers"
EXPANDED_NEED: #1.00 "regarding" #0.45 "tgfb" #0.95 ("tgfb"+"tgf beta"+
    "beta tgf"+"tgfbeta"+"tgfb1"+"tgfb2"+"tgfb3") #0.45 "hnscc" #0.95
    ("hnscc"+"head and neck squamous cell") #1.00 "cancers"
```

Figure 4: Example topic in stage 2 (with query-specific term weights q_T)

created from Medline abstracts. For our experiments, we used only a subset of the AcroMed data. We ignored all acronym/long-form pairs with less than 5 occurrences. The rationale behind this was to avoid low-quality expansion terms caused by wrong AcroMed data. The resulting database contained 25,589 acronyms and 49,822 long forms.

For every input topic, our system searches the topic text for occurrences of acronyms stored in the AcroMed database. In order to allow for lexical variants, an approximate string matching algorithm is used that allows exactly the types of variants that are described in section 4.1.

euGenes

The Genomic Information for Eukaryotic Organisms database (euGenes.org) consists of genetic information for 9 different species. The version used for our experiments contains 184,460 different genes. Our system does not use the information offered by euGenes to its full extent. Instead, it only builds a mapping from gene symbols to full names.

Again, the special needs discussed in section 4.1 are taken into account when the euGenes database is searched for a gene symbol. Furthermore, when looking for expansions, we also allow for narrow synonyms by cutting off the tail of the gene symbol after the last transition from alphabetical to numerical characters or vice versa. The gene symbol "TGFB2" (Human gene Transforming Growth Factor Beta 2), for example, is given the database entries "TGFB2" and "TGFB". The same holds for hyphens, so the gene "ATPsyn-beta" (Drosophila gene ATP Synthase Beta) can even be found when the query is asking for "ATPsyn".

LocusLink

LocusLink is most prominent source of publicly available information on genes. It provides detailed information about the function and position of genes. We used a version of the LocusLink database containing 128,580 entries. Our system does not utilize the full LocusLink information but only the following fields of a LocusLink record:

- `OFFICIAL_SYMBOL`: the officially approved symbol for this gene;
- `PREFERRED_SYMBOL`: a symbol that might become the official symbol in the future, but has not been validated by the nomenclature committee yet;
- `OFFICIAL_GENE_NAME`: the official name of the gene;
- `PREFERRED_GENE_NAME`: similar to the `PREFERRED_SYMBOL` field;
- `ALIAS_SYMBOL`: an alias that is sometimes used to refer to the gene;
- `PRODUCT`: the protein product of the transcript.

A gene can be found in the database by searching for its official symbol, its preferred symbol, one of its aliases, or its protein product. The product is included in this list because genes and their products are often treated as synonyms. Narrow synonyms do not get a special treatment in the LocusLink data because that information in many cases is already included in the `ALIAS_SYMBOL` fields of a gene record.

Some LocusLink records also provide a `PMID` field that contains a list of PubMed articles related to the gene. This information could have been very beneficial if used in an appropriate way. However, it was not clear to us if these fields are of sufficiently high quality and how exactly we could make good use of them.

5 Expansion validation

In stage 2, the results of the first stage are combined using an interleaving fusion technique described by

Yeung et al. [YCC+03]. From each of the three documents lists generated in stage 1, one document is picked in turn and added to the new document list. Duplicates are removed from the result.

The new document list is used to validate the expansion terms in the following way: The system takes the first 150 documents from the result list produced by stage 1 and tries to match them against the expansion terms generated by the expansion techniques described in section 4. Furthermore, it looks for permutations of expansion terms in the documents returned. This is done because the name of the protein that belongs to a certain gene is often a permutation of the gene name. The Homo sapiens gene "glucosidase, alpha; acid", for instance, encodes the "acid alpha-glucosidase" preprotein.

The documents are scanned for the expansion terms or term sequences, and the number of occurrences is counted for every expansion. For every original query term, the best 10 expansions are kept. This set of remaining expansion candidates is then further restricted by removing all expansions that have less than 1% of the occurrences of the top candidate. The remaining expansions are combined into a new disjunctive query element that is added to the original query.

Terms for which an expansion has been found, are given a reduced term weight $q_T = 0.45$. However, the set of validated expansions gets a weight $q_{T'} = 0.95$, so the total weight of the term increases to $q_{T,total} = 1.4$. The idea behind this weight increase is the assumption that terms for which we have an expansion are likely to be key elements of the original query. Having the original term appear twice in the resulting query (once by itself, once in the disjunction) is intended to keep query drift low. Query terms for which no expansion could be found get the default term weight $q_T = 1.0$.

The result of the validation process is shown in figure 4.

6 Pseudo-relevance feedback

In stage 3 of our retrieval system, we expand the query produced in stage 2 by using the results of three difference pseudo-relevance feedback (PRF) algorithms. The output of each PRF process is a vector of term-score pairs. All terms that are already part of the old query (either exactly or in a stem-equivalent form) are removed from the vectors. The results are merged using the normalized CombSUM fusion algorithm [Lee97] [SF94].

From the merged term vector, the best 10 candidates are taken and added to the old query with query-specific term weights

$$q_T = 0.3 \cdot score_T, \tag{5}$$

where $score_T$ is the feedback score of the term T. Because feedback scores are normalized, the top-scoring feedback term will have weight 0.3 in the resulting expanded query.

In the following sections we explain the computation of the feedback scores for each of the three feedback techniques employed.

6.1 Passage feedback

The passage feedback algorithm used by our system creates a vector of expansion candidate terms from a list of scored passages (nodes "MultiText QAP" and "Passage Feedback" in Figure 2). It is based on the QAP passage scoring algorithm described in section 2 and has successfully been used by the MultiText group in TREC 2003 [YCC+03].

The algorithm takes the top 100 documents, as returned by MultiText's QAP, and examines the neighborhood of the best passage within each document. Every time the algorithm encounters a certain term T within that neighborhood, the score of that term is increased:

$$score_T := score_T + \log_2(\frac{N}{f_T}) - \log_2(l), \tag{6}$$

where N is the corpus size, f_T the number of occurrences of T in the corpus, and l is the size of the minimum window that contains both the relevant passage and the term T. If, for example, the relevant passage in the document were "Bart Simpson" and the neighborhood "This is Bart Simpson with his sister Lisa", the size of the minimum window for the term "sister" would be 5. Obviously, this passage feedback algorithm was inspired by the original QAP scoring function.

6.2 Document feedback

The document feedback is similar to the passage feedback. However, it examines entire documents instead of passage neighborhoods. In addition, the document feedback algorithm does not assume that all documents returned by the previous stage are equally relevant. Instead, it gives different weights

to the input documents according to their rank and score in the previous stage.

The algorithm examines the top 100 documents, as returned by BM25 in stage 2. For every term T that appears within a document D, the feedback score of that term is increased:

$$score_T := score_T + w_D \cdot \log_2(\frac{N}{f_T \cdot len_D}), \quad (7)$$

where w_D is the feedback weight of the document D. After trying several document weighting schemes, we decided to use the following formula:

$$w_D = score_D \cdot c^{rank_D}. \quad (8)$$

$score_D$ is the document BM25 document score, $rank_D$ is the document rank in the result list produced by stage 2, and $c < 1$ is chosen in such a way that

$$\sum_{i=1}^{10} c^i = \sum_{i=11}^{100} c^i, \quad (9)$$

i.e., the first 10 documents have the same cumulated weight as the following 90 documents. This weighting function assumes that the results produced in stage 2 are already of high quality.

6.3 Google feedback

While the two feedback techniques described so far rely on the quality of the previous query and the text found in the Medline corpus, our feedback system includes a third technique that uses Google as a source of expansion terms.

The `NEED` part of the original query, stop words removed, is sent to Google. The document snippets returned are considered ordinary documents, and for every term T appearing in a Google snippet S, the term score is updated in the same way as above:

$$score_T := score_T + \log_2(\frac{N}{f_T \cdot len_S}). \quad (10)$$

7 Species names

After the feedback process has finished, a final filtering is performed on the list of documents returned. The filter does not cause any changes to the document ranks unless our system has found one or more species names in the original query.

For example, if the original query contains the terms "human" and "mice", then the filter will try to match the documents against the strings "human", "humans", "homo sapiens", "mouse", "mice", and "mus musculus". The filter is based on a manually created list of 11 species names (and synonyms) that frequently appear in biomedical publications. The idea behind this process is that a species name that appears in a topic is most probably a key word.

The actual filtering is based on the interleave fusion technique. From the list of documents $\mathcal{D}$ returned by BM25, two lists are created:

- $\mathcal{D}_{pos}$ consists of all documents that contain at least one of the search strings;
- $\mathcal{D}_{neg}$ contains all the remaining documents.

$\mathcal{D}_{pos}$ and $\mathcal{D}_{neg}$ are merged into a new document list $\mathcal{D}'$ by taking two documents from $\mathcal{D}_{pos}$ and one document from $\mathcal{D}_{neg}$ in turn. Depending on the relative size of $\mathcal{D}_{pos}$ and $\mathcal{D}_{neg}$, the filtering may leave the document order untouched or change it radically.

8 Evaluation

We submitted two runs for the Genomics track. The first run only made use of the `NEED` field of the given topics. The second run considered `TITLE` and `NEED`. If a query term appeared in both `TITLE` and `NEED`, it was simply given twice the normal term weight ($q_T = 2.0$). The results of our runs are shown in Table 1.

After the runs had been submitted, we found a bug in the validation part of our retrieval system that gave a validation score to the first term in a document n times as high as to the last term, where n is the length of the document. All experiments discussed in this section have been conducted with the corrected version of the validation algorithm. For comparison, we have included both versions (with and without the bug) in Table 1. It can be seen from the table that using the `TITLE` as well improves the quality of the results dramatically over the `NEED`-only run. For the further discussion, we will only consider `TITLE`+`NEED` runs.

8.1 System components

This section deals with the retrieval effectiveness of the individual stages of our system. Table 2 shows the major `trec_eval` values for the intermediate results produced by each stage of our document retrieval system. The first row (*1/BM25,plain*) contains the BM25 baseline, i.e., the results produced

Table 1: Results for NEED-only and TITLE+NEED runs (50 topics, 8,268 relevant documents; "*": bug-fixed versions)

Run	Recall	Avg.prec.	R-Prec.
NEED	5,082	0.3318	0.3630
TITLE+NEED	5,544	0.3867	0.4037
NEED*	5,094	0.3394	0.3716
TITLE+NEED*	5,534	0.3895	0.4046

Table 2: Intermediate results for TITLE+NEED run (default parameters)

Stage/Process	Recall	Avg.prec.	R-Prec.
1/BM25,plain	4,811	0.3353	0.3707
1/BM25,exp.	4,772	0.2775	0.3138
1/QAP,plain	3,892	0.2782	0.3215
2/BM25	5,315	0.3639	0.3845
3/BM25,PRF	5,534	0.3935	0.4108
3/Filtering	5,534	0.3895	0.4046

by an unexpanded query (stop words removed), where documents have been scored by BM25. The last row contains the results for the final document list obtained after species names filtering. We can see that both the number of relevant documents retrived and the mean average precision (MAP) increase significantly, from 4,811 to 5,534 (15.0% improvement) and from 0.3353 to 0.3895 (16.2% improvement), respectively.

From the intermediate values, we can also conclude that the domain-specific query expansion techniques applied in stage 1 and stage 2 constitute a little more than half of this improvement (5,315 relevant documents retrieved, 0.3639 MAP), while the pseudo-relevance feedback performed in stage 3 is responsible for the other half.

Additional experiments have shown that the effectiveness of the second stage is primarily caused by the generation of permutations of query terms/expansions, not by throwing away infrequent expansions.

Perhaps more interesting than the mean average precision, from the point of view of an actual person who has to look at the documents retrieved by our system, is the precision after 20 documents. This value increases from 0.530 (plain BM25 query) to 0.585 (final result), which is a 10.4% improvement.

From Table 2 we can also see that the final species names filtering process could not improve the results in the expected way. We will investigate this phenomenon in section 8.4.

8.2 Domain-specific knowledge

It is clear by now that domain-specific query expansion is beneficial for the effectiveness of our document retrieval system. However, we do not know yet which part of our domain-specific query expansion method is the major factor for the improvement seen. Therefore, we have conducted some additional experiments in which we have selectively disabled certain parts of the query expansion subsystem.

From the results shown in Table 3 (in comparison with Table 2) we see that the general heuristics we employed when generating lexical variants are responsible for the biggest improvement. The second best contributor is the AcroMed acronym database, which causes an improvement of 4.8% over the *Heuristics only* run. It is surprising that adding gene information from euGenes and LocusLink deteriorates the mean average precision (comparing rows *Heuristics&AcroMed* and *All of the above* in Table 3), although the additional data increases the recall from 5,284 to 5,315 relevant documents. We believe that this is mainly because the number of alias symbols provided by the LocusLink database is overwhelming. For the term "TGFB" in topic 14, for instance, the expansion techniques in stage 1 produce 185 candidates (including lexical variants). For "P53" in topic 22, they come up with a set of 176 candidates. This high number of expansion terms entails two possible risks: reduction of the weight of the original query term and query drift.

8.3 Pseudo-relevance feedback

Pseudo-relevance feedback could increase the MAP by 8.1% and the recall by 4.1% over the results produced in stage 2 in our experiments (cf. Table 2). Since we have utilized three different PRF methods, it is interesting to see which method had the greatest impact.

Table 4 shows that passage feedback and document feedback were very effective. We can also see that the combination of both methods (row *Passage&Doc.*) produced even better results than each technique taken alone. On the other hand, Google feedback performed really poorly. It is not clear what caused Google to deliver much worse results than the other feedback methods.

Table 3: Comparison of domain-specific knowledge (results after stage 2)

Technique	Recall	Avg.prec.	R-Prec.
Heuristics only	5,230	0.3550	0.3832
Heu.&AcroMed	5,284	0.3722	0.4059
Heu.&euGenes	5,253	0.3554	0.3827
Heu.&Loc.Link	5,288	0.3631	0.3843
All of the above	5,315	0.3639	0.3845

Table 4: Comparison of pseudo-relevance feedback methods

Method	Recall	Avg.prec.	R-Prec.
Passage PRF	5,470	0.3889	0.4092
Document PRF	5,528	0.3924	0.4065
Google PRF	5,370	0.3708	0.3942
Passage&Doc.	5,559	0.3957	0.4128
All of the above	5,534	0.3935	0.4108

We can, however, identify at least two potential problems related with the use of Google:

- Google only supports exact boolean searches. While this is acceptable for short queries (3-5 terms), it can be problematic when dealing with longer queries.
- Queries are cut off after the 10th term. This does not only mean that we cannot send expanded queries to Google, but even unexpanded queries might be too long. In fact, this was the case for 10 of the 50 topics.

8.4 Species names

The results shown in Table 2 suggest that the final filtering process performed after the pseudo-relevance feedback could not improve the performance of our system and did even decrease its precision. To understand what happened during the filtering process, it is necessary to look at individual topics: Our system could identify species names in 15 out of 50 topics. For 9 of these 15 topics a slight improvement was caused by the filtering, for 4 topics nothing changed, and for the remaining 2 topics the average precision dropped radically (0.8870 to 0.7971 and 0.9478 to 0.7870). Because of the unusually high precision before the filtering, these two topics cannot be considered representative.

Hence, the filtering seems to have a slightly positive impact on the precision for topics with low or medium precision. We did not expect precision values above 70% when designing the filtering algorithm.

8.5 Expansion term weights

In stage 2 and 3, as shown in Figure 4, our system gives a term weight $q_T = 0.45$ to a term T for which we have an expansion. The disjunctive query element (the expansion) that is derived from the original term is given the weight $q_{T'} = 0.95$. Due to the lack of training data, we could not validate the choice of these parameters. Additional experiments conducted after the qrels for the topics had been released showed that by setting $q_T := 0.9$ and $q_{T'} := 0.5$ the MAP could have been increased:

- from 0.3639 to 0.3740 in stage 2;
- from 0.3935 to 0.4042 in stage 3.

This increase, however, carries the cost of a slightly decreased recall (5,253 down from 5,315 documents and 5,519 from 5,534 documents, respectively).

9 Conclusions & Future work

For the TREC 2004 Genomics track, we have implemented a number of different query expansion techniques, based on simple heuristics generating lexical variants of query terms, domain-specific knowledge used to find synonyms and acronym expansions, and pseudo-relevance feedback. We have shown that most of the techniques utilized by our system improved recall and precision.

From the sources we employed for knowledge-based query expansion, the AcroMed database of biomedical acronyms produced expansions of highest quality, outperforming both the euGenes and LocusLink genetic databases.

Our experiments with Google as a source of pseudo-relevance feedback were a little disappointing, since Google performed worse than either of the two other feedback methods. Nonetheless, we will keep using Google for this purpose and try to find a way to circumvent the problems created by Google's restricted query interface. We will also study how our own web corpus, which is about 1 terabyte of publicly available text data, can be used to produce high-quality expansion terms in the special context of biomedical publications.

References

[Acr04] The Medstract Project – AcroMed 1.1. http://medstract.med.tufts.edu/acro1.1/, 2004.

[CCB94] Charles L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. An algebra for structured text search and a framework for its implementation. Technical report, University of Waterloo, August 1994.

[CCL01] Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. Exploiting redundancy in question answering. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 358–276, November 2001.

[CCT00] Charles L. A. Clarke, Gordon V. Cormack, and Elizabeth A. Tudhope. Relevance ranking for one to three term queries. *Information Processing and Management*, 36(2):291–311, 2000.

[euG04] Genomic Information for Eukaryotic Organisms. http://eugenes.org/, 2004.

[Gil02] Donald G. Gilbert. euGenes: A eukaryote genome information system. *Nucleic Acids Research*, 30(1):145–148, 2002.

[Lee97] Joon H. Lee. Analyses of multiple evidence combination. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 267–276. ACM Press, 1997.

[LL04] National Center for Biotechnology iIformation – LocusLink Home Page. http://www.ncbi.nih.gov/LocusLink/, 2004.

[MeS04] U.S. National Library of Medicine – Medical Subject Headings Home Page. http://www.nlm.nih.gov/mesh/, 2004.

[PCC+01] James Pustejovsky, Jose Castano, Brent Cochran, Maciej Kotecki, Michael Morrell, and Anna Rumshisky. Linguistic knowledge extraction from medline: Automatic construction of an acronym database. In *MedInfo 2001: Proceedings of the 10th World Congress on Health and Medical Informatics*, September 2001.

[PM04] U.S. National Library of Medicine – PubMed. http://www.ncbi.nlm.nih.gov/pubmed, 2004.

[RWB98] S. Robertson, S. Walker, and M. Beaulieu. Okapi at trec-7. In *Proceedings of the Seventh Text REtrieval Conference (TREC 1998)*, November 1998.

[RWJ+94] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Proceedings of the Third Text REtrieval Conference (TREC 1994)*, November 1994.

[SF94] Joseph A. Shaw and Edward A. Fox. Combination of multiple searches. In *Proceedings of the Third Text REtrieval Conference (TREC 1994)*, November 1994.

[YCC+03] David L. Yeung, Charles L. A. Clarke, Gordon V. Cormack, Thomas R. Lynam, and Egidio L. Terra. Task-specific query expansion. In *Proceedings of the 12th Text REtrieval Conference (TREC 2003)*, November 2003.