Synonym-based Query Expansion and Boosting-based Re-ranking: A Two-phase Approach for Genomic Information Retrieval

Zhongmin Shi, Baohua Gu, Fred Popowich and **Anoop Sarkar**
School of Computing Science
Simon Fraser University
Burnaby, BC V5A1S6 Canada
{zshi1,bgu,popowich,anoop}@cs.sfu.ca

Abstract

We describe in this paper the design and evaluation of the system built at Simon Fraser University for the TREC 2005 ad-hoc retrieval task in the Genomics track. The main approach taken in our system was to expand synonyms by exploiting a fusion of a set of biomedical and general ontology sources, and apply machine learning and natural language processing techniques to re-rank retrieved documents. In our system, we integrated EntrezGene, HUGO, Eugenes, ARGH, GO, MeSH, UMLSKS and WordNet into a large reference database and then used a conventional Information Retrieval (IR) toolkit, the Lemur toolkit (Lemur, 2005), to build an IR system. In the post-processing phase, we applied a boosting algorithm (Kudo and Matsumoto, 2004) that captured natural language substructures embedded in texts to re-rank the retrieved documents. Experimental results show that the boosting algorithm worked well in cases where a conventional IR system performs poorly, but this re-ranking approach was not robust enough when applied to broad coverage task typically associated with IR.

1 Introduction

The TREC 2005 Genomics track consists of the ad-hoc retrieval task and the categorization task. We were participating in the ad-hoc retrieval task only, due to the considerable effort we spent on building the framework of the biomedical IR system. This is the first time that our team is competing in any TREC task.

The ad-hoc retrieval task aims at the retrieval of MEDLINE records relevant to the official topics. In constrast with the free-form topics of the 2004 task, the 2005 topics are more structured and better defined. A set of 5 generic topic templates (GTTs) was developed following the analysis of the the 2004 topics and the information needs from 25 biologists[1]. Ten topic instances was then derived from each of GTTs. As with last year's ad-hoc retrieval task, the document collection of the 2005 task is a 10-year MEDLINE subset (1994-2003), about 4.6M records and 9.6G bytes in total. The relevance judgement was made by the same pooling method used in the 2004 task, where top ranking documents of every topic from all submitted runs are given to human experts, who then determined each document as definitely relevance (DR), possible relevance (PR) or not relevance (NR) to the topic.

Three run types were accepted this year: automatic, manual and interactive, which differed depending on how the queries were constructed. Each participant was allowed to submit up to two runs. Our submission was in the manual category, since our queries were manually constructed. One of our goals was to determine how natural language processing (NLP) techniques could be used for re-ranking in a post-retrieval step. In our current system, we only apply such techniques for re-ranking. In the future we plan to apply similar techniques towards query expansion.

2 System Architecture

In general, the performance of an IR system largely depends on the quality of the query expansion. Most

[1] http://ir.ohsu.edu/genomics/2005protocol.html

participants of the ad-hoc retrieval task in previous years applied reference database relevance feedback, a technique that finds synonyms and relevant terms from the outside term databases and adds them in the query. Over the past decade, the biomedical databases have evolved dramatically in terms of both the number and the volume, but from the reviews of previous work in this task, most of participants only employed a couple of them to build the reference database. In our system, we collected terms from EntrezGene (EntrezGene, 2005), HUGO (HUGO, 2005), euGenes (euGenes, 2005), ARGH (ARGH, 2005), MeSH (MeSH, 2005), GO (GO, 2005), UMLS (UMLS, 2005) and WordNet (WordNet, 2005), and integrated them into a large reference database which we then use in our system.

Traditional NLP techniques have been generally not successful in improving retrieval performance (Voorhees, 1999), but there is still interest in examining how the linguistic and domain specific knowledge contained in NLP models and algorithms might be applied to specific IR subtasks to improve performance. In this work, we applied a classification technique: a boosting algorithm to capture substructures embedded in texts (Kudo and Matsumoto, 2004) in the second phase of our IR system. Different from the typical bag-of-words approach, the algorithm takes each sentence as a labeled ordered tree and classifies it by assigning a relevance score as either relevant (positive) or not (negative). The relevance of each document is then calculated from relevance scores of the sentences in the document.

Our system consists of two major phases, shown in Figure 1. In the first phase (left to the dashed line in Fig 1), we applied extensive synonym expansion with a conventional IR system, the Lemur toolkit 4.1 (Lemur, 2005). The details of our synonym expansion phase and reference database construction are introduced in §3. The second phase is a post-processing step, in which the boosting classification algorithm (Kudo and Matsumoto, 2004) was used to re-rank the list of retrieved documents from the first phase. §4 describes its implementation details, experiments and evaluations of the boosting-based classification.

The experiments and evaluations in the second phase was not accomplished when we submitted the runs, but we include them in this report and explicitly distinguish them from the submitted results.

3 Conventional IR Module

3.1 Extensive Synonym Expansion

Our system involves the manual selection of key words from the official topics (for most topics the key words were already given in the tabular version of topics) according to the given GTTs. The names and symbols related to each key word, for instance, synonyms, acronyms, hyponyms and similar names, were then matched with the public biomedical and generic databases that include synonyms and relevant terms. Specifically, for gene/protein names, we automatically integrated EntrezGene, HUGO, Eugenes and ARGH into a large gene/protein database with 1,620,947 entries, each of which consists of names and symbols that represent the same biomedical substance, and then matched them with each key word in the topics. Similarly, for diseases, organisms and drugs, related names and symbols were automatically matched with entries in MeSH; molecular functions, biological processes and cellular components made use of GO, and general words/phrases were matched (manually so far) in WordNet. In addition, all sets of related names and symbols were further expanded by searching via the UMLS Knowledge Source (UMLSKS) Socket Server. Figure 2 illustrates the procedure of constructing the reference databases.

3.2 Document Retrieval

In this project, we use the Lemur Language Modeling Toolkit 4.1. The Lemur system was designed to facilitate research in language modeling and information retrieval (IR), such as the ad-hoc and distributed retrieval, structured queries, cross-language document retrieval, summarization, filtering, and categorization.

We use the following three modules provided in Lemur 4.1:

1. Parsing Query module
2. Building index module
3. Structured Query Retrieval Module

In the following subsections, we will briefly describe how each module was used in our system.

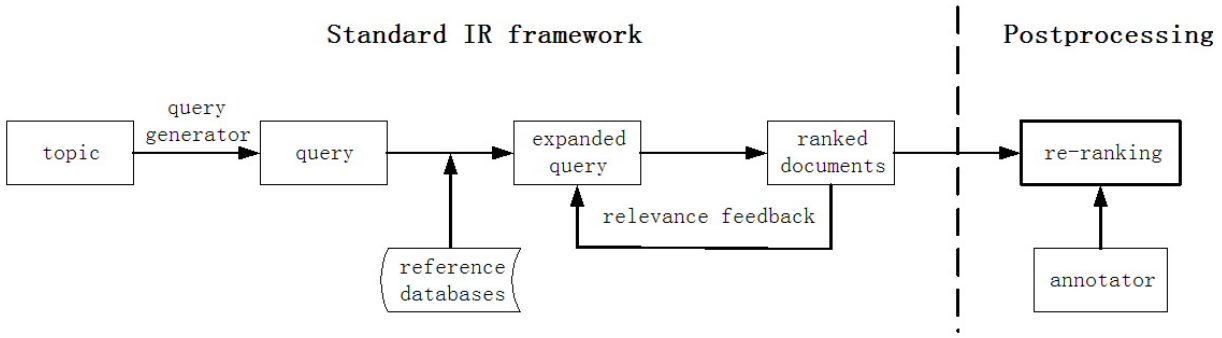

Figure 1: The system architecture

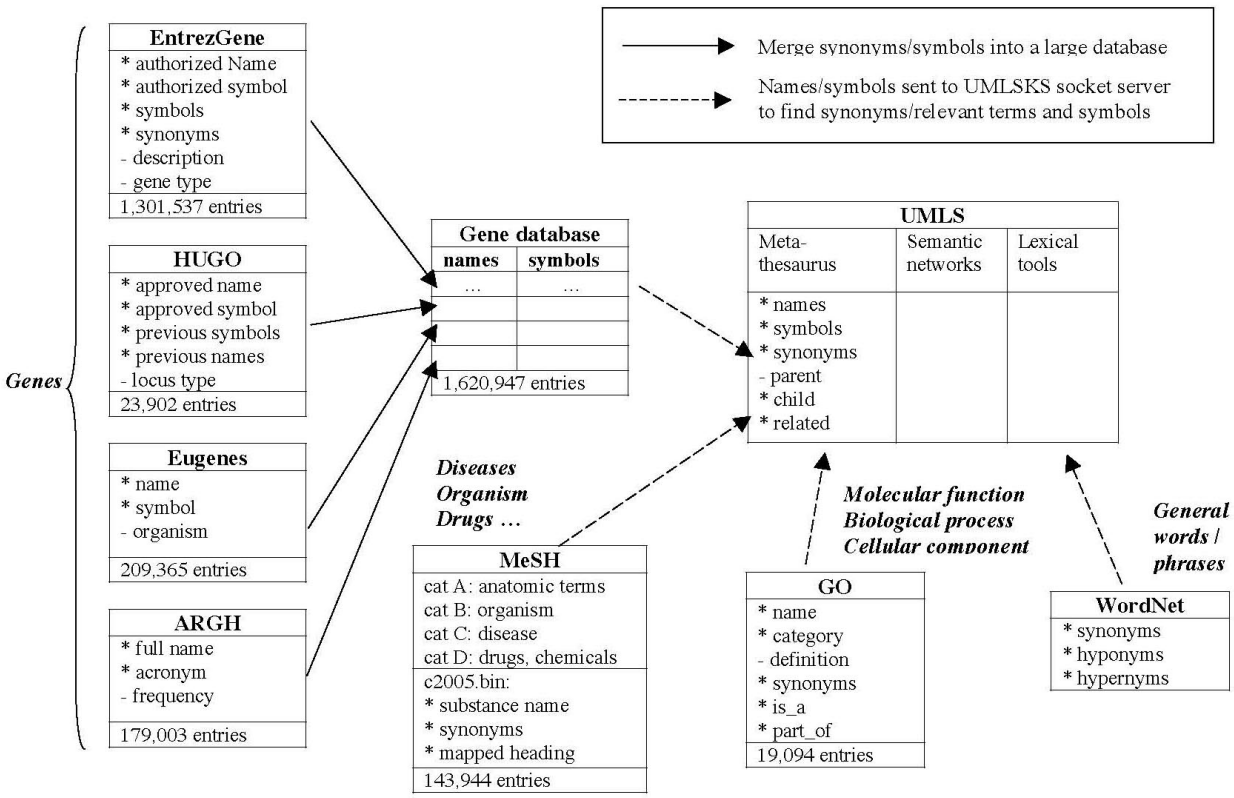

Figure 2: Extensive synonym expansion

3.2.1 Parsing Query

The Parsing Query module contains two utilities to handle different types of queries, ParseQuery and pareInQueryOp. ParseQuery handles queries written in NIST's Web or TREC formats, while ParseInQueryOp is used to parse structured queries written in a structure query language. Both types of queries are then converted into the BasicDocStream format, an document format used inside Lemur. In our experiments, we tried both types of queries and found that the structured queries generally provided better results. Therefore, we used the structured queries in our submitted run.

The structure query language used in Lemur can be found on its web site[2]. Briefly, it allows a user to define various AND/OR/NOT relations, and it provides for weights of sums (WSUM) among the terms. It even allows a user to consider a sequence of single terms by defining them as a phrase. Hence, the structured query enables more precise query definition.

A sample structured query looks like:

$q135 = \#band($
$\#or($
$\#phrase(cellgrowth)$
$\#phrase(cellexpansion)$
$\#phrase(CellularExpansion)$
$\#phrase(CellularGrowth)$
)
$\#or($
$\#phrase(Bop)$
$\#phrase(bacterio-opsin)$
$\#phrase(bacterioopsin)$
$\#phrase(bacterio-opsingene)$
$\#phrase(bop)$
$\#phrase(BiocompatibleOsteoconductivePolymer)$
)
);

3.2.2 Building the Index

Lemur's BuildIndex module supports construction of four types of indices, specifically: InvIndex, InvFPIndex, KeyfileIncIndex, and IndriIndex[3]. We used the KeyfileIncIndex, which includes the position information of a term and can be loaded faster than InvIndex and InvFPIndex while using less disk space than IndriIndex.

3.2.3 Retrieving Structured Query

The structured queries were passed to the StructQueryEval module, which ran retrieval experiments to evaluate the performance of the structured query model using the inquery retrieval method. Note that for structured queries, relevance feedback was implemented as a WSUM of the original query combined with terms selected using the Rocchio implementation of the TFIDF retrieval method (Salton and Buckley, 1988). In our official runs, the parameters (feedbackDocCount,feedbackTermCount, feedbackPosCoeff) for relevance feedback are: 100, 100, and 0.5.

3.3 Evaluation

Among all the official runs submitted to the ad-hoc task of the TREC-2005 Genome Track, 48 are using automatic retrieval methods and 12 including ours are manual ones. Figure 3 shows MAP (upper), P10 (middle) and P100 (lower) scores of the manual runs. Three runs are shown in the figure: the best, the worst and ours on each topic. To better illustrate the performance of our system among others, we plot each value in the figure as the differenece between the actual score and the median score.

Although we do not know the evaluation results of every other system, Figure 3 seems to indicate that our system is above the average. For instance, for the P10 scores of our system on all 49 topics, 36 are above the median and 10 of them are the best; for the MAP scores, 32 are above the median and 2 are the best. The automatic runs perform better than the manual runs on the whole and our system is around the average of the automatic runs. Our future research will involve the invetigation of how our system performs on each topic and each template, looking for insights to further tune our system.

4 Post-processing Module

4.1 Boosting-based Classification

Traditional NLP techniques, such as word sense disambiguation resolution, chunking and parsing, were examined in the IR community at TREC-5 NLP track, but few of them were shown successful for

[2]http://www.lemurproject.org/lemur/StructuredQuery.html
[3]http://www.lemurproject.org/lemur/indexingfaq.html

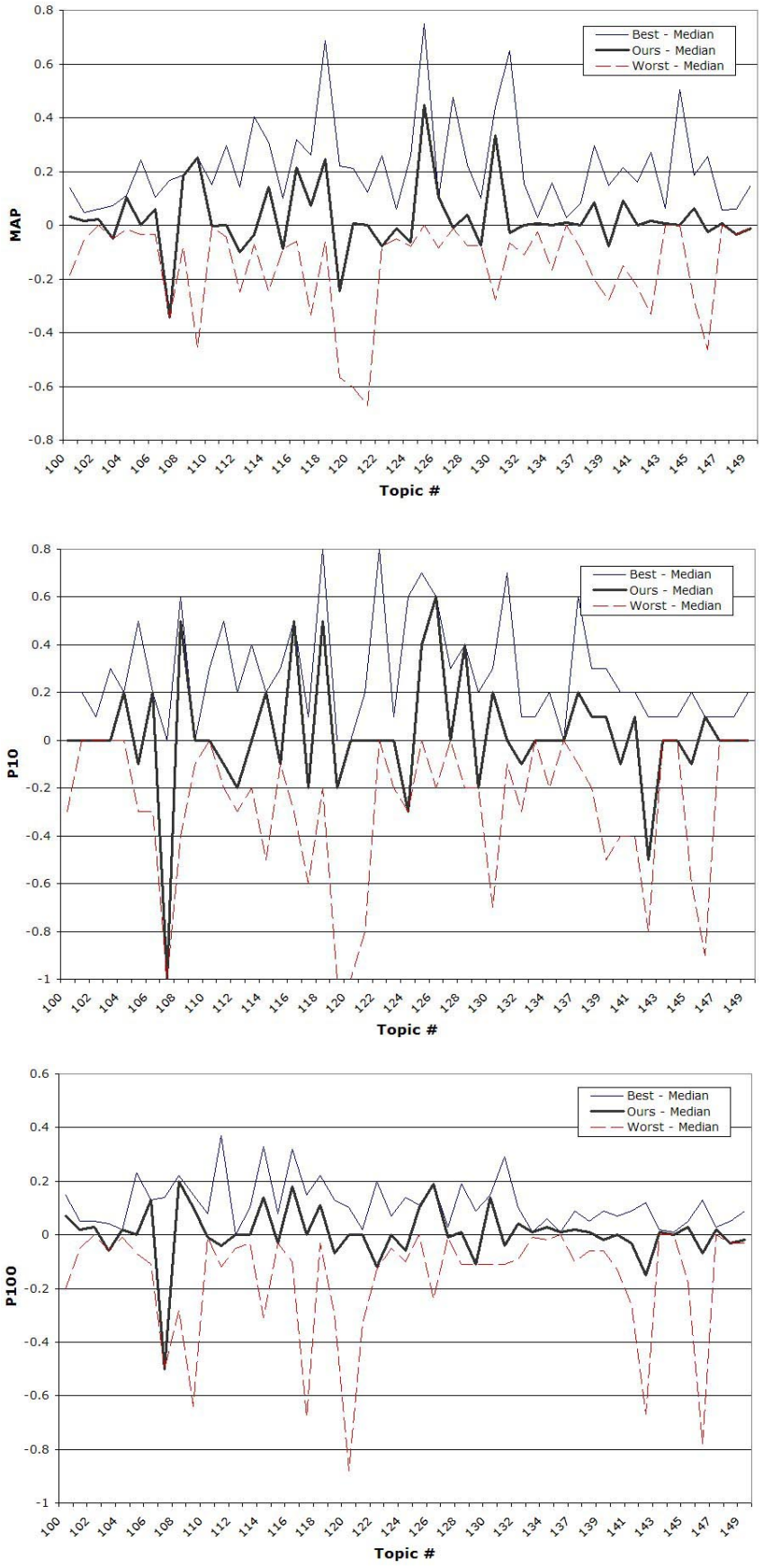

Figure 3: The MAP, P10 and P100 scores of the best, worst manual runs and our system on each topic. Each value is the actual score minus the median.

good retrieval performance. The reasons may lie in the broad coverage of the typical retrieval task, the lack of good weighting schemes for compound index terms and the statistical nature of the NLP techniques (Voorhees, 1999).

However, the attempts of applying NLP and machine learning techniques to the IR tasks are still attractive, since a good understanding of the documents could be a breakthrough to the IR tasks. In this project, we adopted Taku Kudo's Boosting Algorithm for Classification of Trees (BACT), a classification method that captures the sub-structures embedded in texts. We use the method and implementation described in (Kudo and Matsumoto, 2004). BACT takes a set of all subtrees as the feature set, from which it iteratively calls a weak learner to produce weak hypotheses. The strong hypothesis is finally generated by a linear combination of weak hypotheses.

We incorporated BACT into the post-processing step, where the list of retrieved documents from Lemur was re-ranked by taking the classification of the documents into account, as shown in the Figure 4. The documents in the training data were parsed using Charniak's parser (Charniak, 2000) and then classified by BACT in terms of relevant (positive) or irrelevant (negative). A re-ranking mechanism made the final relevance decision by combining the relevance scores from both Lemur and BACT.

The major difficulty of applying BACT in this task is that it assigns a classification score (positive or negative) to each sentence rather than assigning a score to a document. This resulted in two issues: 1) the lack of the training data with the label for each sentence; 2) the lack of a mechanism for combining sentence scores into a document score.

Since we lacked training data of sufficient quality and quantity for the classification task, we were not able to submit the post-processing results to the TREC in time for the initial deadline. After the results of the ad-hoc retrieval task were announced (on Sept. 30, 2005), we were able to test the performance of the post-processing, by taking the following steps to prepare the training and test data for BACT:

1. The retrieved documents in the first two topics in each TTL were taken as the test data and those in the remaining topics as the training data.

2. The irrelevant documents in the training data were removed due to the unbalance of the training data (irrelevant documents are much more common than relevant ones).

3. In the training data, sentences were given "approximate" labels by matching them against a disjunction of all terms in the corresponding query as either matched $(+1)$ or unmatched (-1).

BACT assigned a real number as the classification score to each sentence, with a larger score corresponding to a more relevant the sentence. We took the mean of all sentence scores in each document as the document score.

4.2 Re-ranking

The goal of re-ranking is to combine R_L and R_B, the ranks from Lemur and BACT respectively, such that the rank R' maximizes the evaluation scores, for example, MAP, P10 and P100. R_L, R_B and R' are score vectors of retrieved documents. We assumed that such a combination was linear, i.e.:

$$R' = R_L + i * R_B \tag{1}$$

We thus looked for i' that maximizes the evaluation function $E(R')$:

$$i' = \text{argmax}_i \, E(R_L + i * R_B) \tag{2}$$

4.3 Evaluation

As described in §4.1, we extract relevance scores of our retrieved documents (by Lemur) from the evaluation results of 2005 ad-hoc retrieval task. For each TTL, the retrieved documents of the first two topics were taken as the test data, and those of the remaining topics as the training data.

Table 1, 2 and 3 list the MAP, P10 and P100 before $(i = 0)$ and after the re-ranking for the TTL #1, #2 and #3. A linear combination coefficient i' was predicted for each TTL following Equation 2. For the TTL #2, i' converges at 15 and the linear combination model significantly improves the IR performance: MAP increases from 0.0012 to 0.0024 for

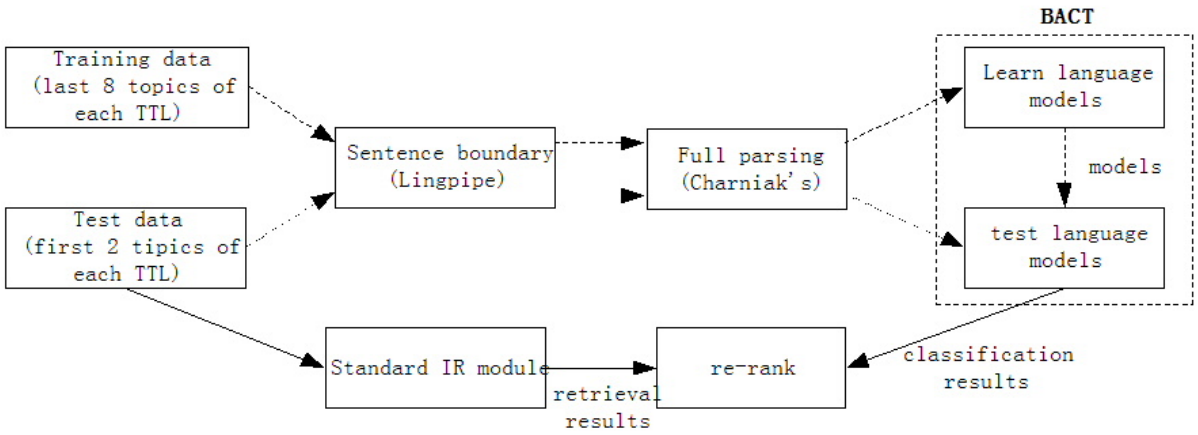

Figure 4: The post-processing phase

Topic #	Metrics	$i = 0(i')$	$i = 10$
100	MAP	0.2221	0.1785
	bpref	0.8649	0.8649
	P10	0.4	0.3
	P100	0.28	0.22
101	MAP	0.0685	0.0195
	bpref	0.75	0.75
	P10	0	0
	P100	0.07	0.01

Table 1: Performances of re-ranking on the TTL #1

Topic #	Metrics	$i = 0$	$i = 15(i')$
110	MAP	0.0012	0.0024
	bpref	0.25	0.25
	P10	0	0
	P100	0	0.01
111	MAP	0.0492	0.1602
	bpref	0.4356	0.4356
	P10	0.1	0.7
	P100	0.1	0.4

Table 2: Performances of re-ranking on the TTL #2

the topic #110 and from 0.0492 to 0.1602 for the topic #111; Same situations for P10 and P100. However, for the TTL #1 and #3, no linear combination model can improve the IR performance, i.e., $i' = 0$. The scores at $i = 10$ are also listed in Table 1 and 3 to show that the performance dropped when the linear combination models were applied.

4.4 Discussion

Our experiments show that BACT as the post-processing does help when *bpref* (proportion of judged relevant documents that are retrieved) of the conventional IR system is low, for instance, 0.25 and 0.4356 in the TTL #2. For the TTL #1 and #3 where BACT failed, the average *bpref* is very high, above 0.8.

It seems as if our current use of BACT for re-ranking cannot scale to the broad coverage of relevant documents in the retrieved document set, especially in the case where *bpref* is high. This is a

Topic #	Metrics	$i = 0(i')$	$i = 10$
120	MAP	0.6113	0.2410
	bpref	0.8145	0.8145
	P10	1	0.3
	P100	0.88	0.29
121	MAP	0.6697	0.0328
	bpref	0.8810	0.8810
	P10	0.8	0
	P100	0.34	0

Table 3: Performances of re-ranking on the TTL #3

common problem of NLP techniques when applied to the IR task. However, employing machine learning and NLP techniques such as BACT as the post-processing step may help the conventional IR system when the recall is low, by re-ranking the retrieved documents towards a better performance.

Acknowledgments

We thank Yang Wendy Wang for the helpful discussions and the constructive comments during the system design and experiments. We greatly acknowledge the organization committee of the TREC 2005 genomics track and those who participated in the topic generation and relevance judgment for their valuable work.

References

ARGH. 2005. *Biomedical Acronym Resolver*. http://invention.swmed.edu/argh/.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Meeting of the North American Chapter of the ACL*, pages 132–139.

EntrezGene. 2005. *National Center of Biotechnology Ingormation*. http://www.ncbi.nlm.nih.gov/entrez/query.fagi?db=gene.

euGenes. 2005. *Genomic Information for Eukaryotic Organisms*. http://eugenes.org/.

GO. 2005. *The Gene Ontology*. http://www.geneontology.org/.

HUGO. 2005. *Gene Nomenclature Committee*. http://www.gene.ucl.ac.uk/nomenclature/.

Taku Kudo and Yuji Matsumoto. 2004. A boosting algorithm for classification of semi-structured text. In *Proceedings of Empirical Methods of Natural Language Processing (EMNLP 2004)*.

Lemur. 2005. *Language Modeling Toolkit 4.1*. http://www.lemurproject.org/lemur/doc.html.

MeSH. 2005. *Medical Subject Headings*. http://www.nlm.nih.gov/mesh/meshhome.html.

G. Salton and C. Buckley. 1988. Term eighting approaches in automatic text retrieval. *Information Processing and Management*.

UMLS. 2005. *Unified Medical Language System*. Ntional Institute of Health, United States, http://www.nlm.nih.gov/research/umls/.

Ellen M. Voorhees. 1999. Natural language processing and information retrieval. In *SCIE*, pages 32–48.

WordNet. 2005. *A lexicon database for English Language*. http://www.cogsci.princeton.edu/wn/.