TREC 2005 Genomics Track Experiments at IBM Watson

Rie Kubota Ando
IBM Watson Research
Hawthorne, NY, USA
`rie1@us.ibm.com`

Mark Dredze [*]
University of Pennsylvania
Philadelphia, PA, USA
`mdredze@seas.upenn.edu`

Tong Zhang [†]
Yahoo Inc
New York City, USA
`tzhang@yahoo-inc.com`

Abstract

This paper describes our experiments in the TREC 2005 Genomics Track. For the ad-hoc retrieval task, we study synonym-based query expansion, as well as the effectiveness of a new pseudo-relevance feedback method which is derived from our recent work on semi-supervised learning. For the categorization task, we study various methods for estimating conditional class probability and determining the optimal threshold parameter — essential for obtaining high performance results for this task.

1 Introduction

This paper reports on our participation in the TREC 2005 Genomics Track. The submissions were on two tasks: ad-hoc retrieval and categorization. The goal of the ad-hoc retrieval task was to search a Medline corpus consisting of biomedical paper abstracts. We experimented with a new pseudo-relevance feedback method derived from a recently proposed semi-supervised learning method, as well as domain database synonym lookup for query expansion. The goal of the categorization task was to select journal articles to be cataloged in the Mouse Genome Informatics database (MGI). On this task, we used a regularized linear classifier and experimented with supervised and semi-supervised learning approaches, with emphasis on class probability estimation-based threshold determination methods.

Our systems are competitive on both tasks. On the ad-hoc retrieval task, our two official runs produced the second and third best map results among all the 45 automatic runs, with small differences from the top automatic run. On the categorization task, our official run achieved the best utility on one of four sub-tasks.

The paper is organized as follows. We present our ad-hoc retrieval system in Sections 2.1–2.4. The results on the 2004 and 2005 topic sets are reported in Section 2.5. The categorization systems and the performance results are presented in Section 3.

2 Ad-hoc retrieval

It appears from the literature that successful systems in the 2004 Genomics Track are typically equipped with some form of pseudo relevance feedback and query expansion using domain databases. We adopt this framework and focus on:

- Developing a new automated feedback method, which we will refer to as *structural feedback* .
- The use of domain databases for obtaining synonyms, and their use for query expansion.

The high-level data flow is shown in Figure 1. We first index documents in the corpus, which we view as the generation of document vectors. For a given search topic, our system generates a query vector incorporating synonyms for query terms using domain database lookup. For the generation of both query and document vectors, we use term weighting similar to the popular

[*]This work was done when the second author was a summer visitor at IBM T.J. Watson Research Center.

[†]This work was done when the third author was at IBM T.J. Watson Research Center.

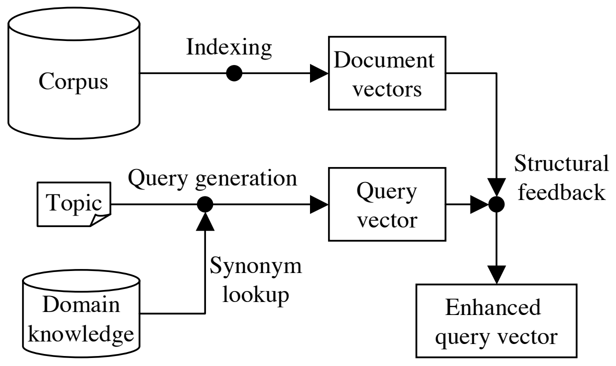

Figure 1: Ad-hoc retrieval data flow (high-level).

BM25 (Robertson et al., 1994). A search of the index given the query vector is performed by taking the inner product of the query vector and each document vector. From the query vector and document vectors thus retrieved, structural feedback generates a new enhanced query vector. The final search results are obtained by taking inner products of this new query vector and document vectors.

We first present the structural feedback method (Section 2.1), and then describe the use of domain knowledge for enhancing queries in Section 2.2. The 2005 topic set differs from the 2004 topic set in its introduction of five topic templates. We discuss the implications of templates in Section 2.3. After describing the implementation details such as term weighting in Section 2.4, we report performance results on the 2004 and 2005 topic sets (Section 2.5).

2.1 Using unlabeled data through structural feedback

This section describes our new automatic feedback algorithm. The idea is to learn a relevant structure shared by multiple variations of the original query, and then use this shared structure to improve IR performance. The development of our new automatic feedback method stems from the *Alternating Structure Optimization (ASO)* algorithm recently proposed for semi-supervised learning.

2.1.1 ASO and its application to semi-supervised learning

ASO is a machine learning algorithm designed to improve prediction performance (e.g. classification performance) by simultaneously learning multiple prediction problems that are related to each other (Ando and Zhang, 2005a; Ando and Zhang, 2005b). The shared structure is learned by joint empirical risk minimization over these multiple related tasks. The learned structure is then transferred to the target prediction problem of interest.

The intuition is that by observing many related problems, one can learn useful information (predictive structure) shared by these problems and then use it for improving overall performance. In particular, the application of ASO in the semi-supervised setting exploits unlabeled data by creating multiple prediction problems and their labeled examples automatically from unlabeled data. If these created prediction problems (called *auxiliary problems*) are related to the target prediction problem of interest, then the shared predictive structure learned by ASO will be useful for the target problem.

We do not include the algorithmic details of ASO in this paper; instead, we refer the reader to (Ando and Zhang, 2005a; Ando and Zhang, 2005b). Here we simply state that ASO produces a structure matrix Θ, which captures information learned from unlabeled data via auxiliary problems. The rows of Θ are essentially the most significant left singular vectors of a matrix of feature-weight vectors learned on auxiliary problems. (An alternating optimization procedure can be used to improve the learned structure Θ, which gives the name of the method). From Θ and a feature vector x, we obtain a new feature vector representation:

$$\hat{x} = \begin{bmatrix} x \\ \Theta x \end{bmatrix} . \tag{1}$$

The final step of ASO is to perform training for the target problem using the new feature vector representation in (1) and the labeled data. It has been shown that the inclusion of unlabeled data in this way (i.e., via Θ in (1)) improves prediction performance on a number of tasks such as text categorization, named entity and syntactic phrase recognition.

2.1.2 Structural feedback overview

Suppose that vector representations (or a feature mapping) of a query and a collection of documents are given as input. Essentially, our structural feedback algorithm employs a structural learning idea similar to that of ASO. Specifically, we consider search as a problem of predicting a document's relevancy/irrelevancy to the topic (represented by the query), using a linear prediction model. We further consider the given query vector as the only positive ('relevant') example available for training on this task, while regarding all the documents (that are to be retrieved) as unlabeled data.

From this viewpoint, negative ('irrelevant') examples are not explicitly given. Therefore instead of using a discriminative learning method that tries to separate positive data from negative data, it is more appropriate to consider generative learning methods where a model for each class can be constructed from positive data only. Such generative models include Naive Bayes and Centroid methods (which uses the mean of the positive data points as the weight vector). Because a weighting scheme similar to BM25 has been applied priori, we shall simply adopt the centroid method in our approach. With only the query as positive data, the weight vector after training (using a centroid method) trivially results in the query vector itself. Since we assume a linear model, the prediction or output value (relevancy score) is given by the inner product of the weight vector (query vector) and a feature vector (document vector). This is consistent with traditional IR methods.

We know from the machine learning literature that one can improve prediction performance by using unlabeled examples in addition to labeled examples – this is essentially the semi-supervised learning problem. A natural question to ask is whether related ideas can be applied to information retrieval, when posed as a learning problem such as outlined above. In this case, unlabeled examples are simply documents to be retrieved since their labels (relevancy/irrelevancy to the search topic) are not given.

Recall that the essence of ASO is to learn a useful predictive structure from multiple related problems. To apply this idea to the task of relevancy prediction in IR, we create auxiliary problems by generating a number of variants of the given query. More precisely, we consider the documents 'highly ranked' by these query variants as the positive examples of the auxiliary prediction problems, and apply structural learning to these auxiliary problems. Our method then generates the structure matrix Θ, which captures predictive structure shared by the auxiliary problems. If we apply the original ASO formulation, new feature vectors (a new query vector and new document vectors) would be generated as in (1). However, for efficiency, we keep document vectors as they are and only change the query vector by:

$$\hat{q} = q + \Theta^T \Theta q \ . \qquad (2)$$

It is easy to verify that doing so is equivalent to (1) for our purpose. Thus, we obtain a new enhanced query vector $\hat{q}$.

```
Input: initial query vector $q$
Parameters: $m$, $h$
Output: new query vector $\hat{q}$
Generate query variants $q_1, q_2, \cdots, q_p$ from $q$.
$S := \{$ $m$ documents highly ranked by query $q$ $\}$
for $i = 1$ to $p$
  $S_i := S \cap \{$ documents highly ranked by $q_i$ $\}$
  $w_i := \sum_{d \in S_i} d / \| \sum_{d \in S_i} d \|_2$
End for
Let $W$ be a matrix whose $i$-th column is $w_i$.
Set the rows of $\Theta$ to be $W$'s $h$ most significant
  left singular vectors.
$\hat{q} := q + \Theta^T \Theta q$
```

Figure 2: Overview of the structural feedback method

2.1.3 Structural feedback implementation details

Although there are several ways to generate useful query variants, our implementation generates variants by removing up to k tokens (for $k = 0, 1, 2$) from the input query. For instance, if the given query consists of { ferroportin, 1, iron, human }, we will have eleven query variants: four three-token variants ({ ferroportin, 1, iron }, { ferroportin, 1, human }, { ferroportin, iron, human }, { 1, iron, human }); six two-token variants ({ ferroportin, 1 }, { ferroportin, iron }, $\cdots$); and the original query.

To retrieve documents by these query variants, it is too expensive to search the entire corpus. Instead, we first retrieve m documents by the given query, and from these m documents we choose the ones highly ranked by the query variants. This requires the use of a cut-off criterion for choosing highly-ranked documents. Consider a pseudo document in which every token in the query variant q_i occurs only once, and let $\tilde{d}_i$ be the document vector (generated as in Section 2.4.2) for this pseudo document. We say that document (vector) d is *highly ranked* by q_i if and only if $q_i^T d \geq q_i^T \tilde{d}_i$.

In the original ASO algorithm, empirical risk minimization is used for training discriminative classifiers on the auxiliary problems. As mentioned earlier, due to the absence of negative examples, we employ a generative learning model in this work. Specifically we adopt the (normalized) centroid method where the feature-weight vector w_i is given by the length-normalized average of the positive examples (highly ranked document vectors that are selected). Then, as in ASO, we construct a matrix W so that its i-th column is the weight vector w_i and compute W's h most significant left singular vectors, which give the rows of the structure matrix Θ. In our experiments we set the dimension parameter $h = 1$.

Our implementation of the structural feedback method has two additional parameters. One is m, the number of documents retrieved by the initial query. In our official runs, we set $m = 30$. The performance is relatively insensitive to m as long as m is not too small. The second parameter is the number of terms we keep in the new query vector $\hat{q}$. Although it might be ideal, using $\hat{q}$ as is considerably increases the search runtime. We zero out all but the k largest entries of $\hat{q}$. The performance seems to be relatively insensitive to k in the range of $k \geq 50$. In our official runs, we set $k = 100$.

Figure 2 summarizes the structural feedback method.

2.1.4 Discussion

Apart from the theoretical justification given in the original ASO work (Ando and Zhang, 2005a), the intuitive meaning of structural feedback becomes clearer when we set $h = 1$. The structure matrix Θ for $h = 1$ has only one row, for which we write θ_1 so that $\theta_1 = \Theta^T$. By construction, θ_1 is the most significant left singular vector of the auxiliary feature-weight matrix. Therefore, for p query variants, we have:

$$\theta_1 = \arg\max_{|\theta|=1} \sum_{i=1}^{p} (\theta^T w_i)^2 \ , \qquad (3)$$

$$\hat{q} = q + (\theta_1^T q)\theta_1 \ . \qquad (4)$$

That is, the new query vector $\hat{q}$ is a weighted sum of the original query q and an additional query vector θ_1. In the additional query vector θ_1, query terms are automatically weighted based on how representative they are in the documents highly ranked by query variants, as seen from (3). The coefficient $\theta_1^T q$ serves as a global weight of the additional query θ_1. This global weight becomes relatively large, if the original query terms are relatively representative in the documents retrieved by many of the query variants. As such, $\theta_1^T q$ serves as an automatic weight that reflects confidence in the usefulness of the additional query θ_1. That is, if queries slightly different from one another retrieve essentially similar documents in which the original query terms relatively dominate, one could more confidently say that the terms from those retrieved documents should be useful as query terms.

2.2 Using domain knowledge

2.2.1 Database lookup for synonyms

It is well known that the key concepts in this domain (such as genes and proteins) have many aliases. We use the following databases to look for synonyms:

- LocusLink[1]
- Gene Ontology (GO)[2]

[1] http://www.indiana.edu/˜rac/clsd/locuslink.html
[2] http://www.geneontology.org

- Mesh[3]
- Swiss-Prot[4]

We search the databases for longest match with the given topic text and add the synonyms listed in the matched database entries to the query.

In addition, we use an abbreviation lexicon to map abbreviations to their full forms (e.g., mapping "TGFB" to "Transforming Growth Factor Beta"). We automatically generated this lexicon from the Medline corpus by a method essentially similar to (Schwartz and Hearst, 2003). To improve the mapping precision we remove entries that occurred less than five times and select the most frequent full form if multiple forms exist.

Given the initial query vector q, our new query vector q_s enhanced by the database lookup is: $q_s = q + 0.1s$, where s is the query vector generated (as in Section 2.4.2) from the additional tokens from the synonyms and the full forms of abbreviations.

To apply structural feedback to the query enhanced with synonyms, we use q_s to initially retrieve the m documents, but we use the original query q to generate query variants. The final query is then generated by $q_s + (\theta_1^T q_s)\theta_1$.

2.2.2 Bi-grams

Biochemical entity names often contain both alphabetical characters and digits and have notational variations (e.g., "ABC1" vs. "ABC-1"). To counteract this, we add selected token bi-grams to the index and queries. More specifically, we generate token bi-grams (e.g., "ABC_1" where "_" indicates the token boundary) whenever we see an alphabetic chunk followed by a digit chunk with/without a white space or a hyphen between them (e.g., "ABC1", "ABC 1", "ABC-1"). Similarly, we generate token bi-grams (e.g., "1_ABC") for digit chunks followed by alphabetic chunks (e.g., "1ABC", "1 ABC", "1-ABC"). This is done both in indexing and in query generation. The token bi-grams added to the queries and documents are treated in the same way as the other tokens in the generation of query/document vectors.

[3] http://www.nlm.nih.gov/mesh/meshhome.html

[4] http://www.ebi.ac.uk/swissprot/

2.3 Topic templates

Since the Genomics Track is relatively new, the only available topic set with relevance judgments useful for the development of our methods is the one from 2004. Although the 2004 topics are in the usual TREC format, the 2005 topics are expressed in five topic templates. Most of these templates have two slots such as "roles of gene X in disease Y". The implication is that the 2005 topics look for more specific information than the 2004 topics (e.g., "information on gene X"); consequently, the 2005 topics are apparently more difficult than the 2004 topics. (When using two-slot topic templates, finding X only is not enough. Relevant documents are required to contain both an X and a Y that are related to each other in the designated way.) In addition, the 2005 topics are substantially less verbose than the 2004 topics, which have the title and need sections. Such terse topics (conveying less information) pose a challenge.

If a sufficient amount of development data were provided, we could have worked on developing methods that address such issues. However, only two sample topics per template were made available, and their relevance judgments were indicated as incomplete/unreliable. Given the absence of appropriate development data, we decided not to do anything special with the newly introduced topic format.

2.4 Implementation details

2.4.1 Indexing

We tokenize the text at white spaces after replacing all the non-alphabetic and non-digit characters with white spaces. The Porter stemmer is used for stemming, and stopwords (function words) are removed. In this manner, we obtain tokens (or terms) from the TI (title), AB (abstract), and RN sections of the Medline articles, and index documents with these terms. Our experiments use an in-house search engine based on the conventional inverted file mechanism.

descriptions	map	
baseline (simple queries)	37.95	–
bi-gram	39.56	(+1.61)
synonyms	39.98	(+2.03)
(2004 best (Fujita, 2004))	40.75	(+2.80)
bi-gram + synonyms	41.45	(+3.50)
structural feedback	44.20	(+6.25)
structural feedback + bi-grams	45.19	(+7.24)
structural feedback + bi-grams + synonyms	45.52	(+7.57)

Figure 3: Mean average precision results (%) on 2004 topics. The numbers in parentheses are the gains compared to our baseline performance.

2.4.2 Query and document vector representations

A query is generated from the narrative-format topic text, removing stopwords and the template words (e.g., "role"). We generate a query vector q and document vector d by setting the j-th entries (corresponding to the j-th term) $q[j]$ and $d[j]$ to:

$$\begin{aligned} q[j] &= \sqrt{\mathrm{idf}_j} \cdot \frac{f_j \cdot (k_3 + 1)}{f_j + k_3} \ , \\ d[j] &= \sqrt{\mathrm{idf}_j} \cdot \frac{f'_j \cdot k_1}{f'_j + k1\left((1-b) + b \cdot l/\bar{l}\right)} \ , \\ \mathrm{idf}_j &= \log\left(\frac{n+1}{\mathrm{df}_j + 0.5}\right) \ , \end{aligned}$$

where f_j and f'_j are the frequencies of the j-th term in the topic and document, respectively, l is the length of document, $\bar{l}$ is the average length of the documents, df_j is the document frequency of the j-th term, and n is the number of documents. The above feature weighting is essentially the same as BM25 (Robertson et al., 1994). Although IR term weighting might be typically described as a product of $q[j]$ and $d[j]$, we instead consider a query vector and a document vector separately as feature vectors. Throughout the experiments, we set $k_1 = 1.2, k_3 = 7, b = 0.75$, adopted from the Lemur[5] default setting.

[5]http://www.lemurproject.org

2.5 Ad-hoc retrieval results

2.5.1 Results on the 2004 topics

Figure 3 shows the mean average precision (map) results on the 2004 topics. Our baseline uses queries generated without structural feedback or synonyms/bi-grams. The TITLE and NEED sections were used to generate the initial query vectors.

Most notably, structural feedback greatly improves performance over the baseline, producing 44.20% map. Combining structural feedback with bi-grams and synonyms, the map is 45.52%, which is 7.57% higher than our baseline and 4.77% higher than the 2004 top system (Fujita, 2004).

Bi-grams and synonyms are both effective on the 2004 topics. However, upon the inspection of the performance on individual topics, we found that bi-gram queries are somewhat unstable – i.e., bi-grams greatly improve performance on some topics but significantly degrade performance on other topics. Therefore, we decided to submit one run with "structural feedback+synonyms" and the other run with "structural feedback+bi-grams+synonyms".

2.5.2 Results on the 2005 topics

Figure 4 shows the mean average precision results on the 2005 topics. Overall, the performance is lower than on the 2004 topics. This is not surprising. As mentioned in Section 2.3, the 2005 topics are apparently more challenging, presumably as a result of introducing templates.

It turned out that unlike the 2004 topics, syn-

run names	descriptions	map	
–	synonyms	25.89	(−0.21)
–	baseline	26.10	–
–	bi-grams	26.54	(+0.44)
ibmadz05bs	structural feedback +synonyms+bi-grams	28.59	(+2.49)
ibmadz05us	structural feedback +synonyms	28.83	(+2.73)
(york05gm1)	(best automatic run)	28.88	–
–	structural feedback +bi-grams	29.04	(+2.94)
–	structural feedback	29.74	(+3.64)
–	structural feedback w/ m=20	30.10	(+4.00)
–	structural feedback w/ m=10	30.16	(+4.06)
(york05ga1)	(best manual run)	30.20	–

Figure 4: Mean average precision results on 2005 topics. The parameter m for structural feedback was set to 30 unless specified otherwise. The numbers in parentheses are the gains compared to our baseline performance.

onyms which we obtained from several domain specific databases slightly degraded performance on the 2005 topics. Bi-grams improve performance by 0.44% when we do not use the structural feedback method. However, our finding is that, in fact, the structural feedback method alone achieves higher performance than combinations of structural feedback with synonyms and/or bi-grams (which are our official runs).

The map result of our best official run (ibmadz05us) is the second best among all the automatic runs, with a small difference (0.05%) from the best run. Using structural feedback alone, we are able to produce a map 30.16%, which is higher than the best automatic run and very close to the best manual run.

It is encouraging to confirm that the new structural feedback method consistently performs well. Moreover, in our private experiments not included in this report, we observed that this new idea outperformed some conventional pseudo-relevance feedback methods which we also implemented. We believe that this new approach warrants further investigation.

2.5.3 Discussion

On the ad-hoc retrieval task, we experimented with a new automatic feedback method which we call structural feedback, as well as query expansion using synonyms found in domain specific databases. The former turned out to be consistently useful, while the usefulness of the latter is inconclusive.

The potential danger in adding synonyms from databases appears to be at least two-fold. First, we may introduce irrelevant query terms because of 'noisy' database entries or the inherent ambiguity of the matched entries. Secondly, and more importantly, even if we can correctly obtain synonyms, it is not clear, at least to us, what term weights should be assigned to the tokens resulting from those synonyms (which are often multi-word expressions). If the weights assigned to synonym tokens are too large, they may, in effect, 'dominate' other terms that happen to have no synonyms. Such imbalance of term weights is a critical issue, especially when relevant documents must contain two concepts with a specific relationship.

On the other hand, structural feedback, in effect, automatically adjusts for each topic, the weight of the additional query vector θ_1, based on an implicitly estimated 'confidence'. Our experience indicates that the resulting method is quite effective. It might be useful to investigate a similar 'fail-safe mechanism' for synonyms from external resources.

Our submission system is purely based on general purpose search technology, without any specific components for topic templates and the semantical meanings they imply. Although we believe a more dedicated retrieval system specialized to such applications could yield better performance on this particular TREC task, it is encouraging to see that our system, which is not designed specifically for this track, produces good results with the help of our new structural feedback method.

3 Categorization

The purpose of the Genomics Track categorization task is to triage MGI journal articles according to one of four information needs. There are four sub-tasks corresponding to the information sought: A (Alleles of mutant phenotypes), G (Gene Ontology annotation), T (Tumor biology), and E (Embryologic gene expression). We submitted three runs for each.

3.1 Measurement

The task setting differs from typical text categorization tasks in that the evaluation metric, so-called *utility*, is specifically designed for this track, which penalizes false negatives far more severely than false positives.

The version used in the evaluation is a normalized utility defined as $U_{norm} = U_{raw}/U_{max}$. For a test collection of documents to categorize, U_{raw} is calculated as follows: $U_{raw} = (u_r * TP) + (u_{nr} * FP)$, where u_r is the relative utility of relevant documents, and u_{nr} is the relative utility of non-relevant documents. TP is the true positive and FP is the false positive. U_{max} is the maximum possible utility under perfect classification. With $u_{nr} = -1$, the values of u_r used in the evaluation are 17 for A, 64 for E, 11 for G, and 231 for T.

To maximize utility, special attention is needed to determine the cut-off threshold. Suppose that the classification method can estimate the conditional in-class probability $P(Y = 1|X)$, and that we retrieve all documents X with $P(Y = 1|X) \geq \theta$ for some cutoff threshold θ. Since the probability of X being TP is $P(Y = 1|X)$, and being FP is $1 - P(Y = 1|X)$, any document above the optimal threshold θ should contribute to U_{raw} positively on average. This implies that we should choose the threshold such that $u_r\theta + u_{nr}(1-\theta) = 0$. That is, we should retrieve every document X such that

$$P(Y = 1|X) \geq \theta = -u_{nr}/(u_r - u_{nr}). \qquad (5)$$

This method of threshold selection is also used in (Dayanik et al., 2004).

3.2 Learning methods

We employ regularized linear classification with modified Huber loss and square (and optionally 1-norm) regularization, separately for each task. Let the training data be (X_i, Y_i) $(i = 1, \ldots, n)$, where X_i is the vector representation of a document, and $Y_i \in \{\pm 1\}$ be the corresponding label. We seek a linear weight vector $\hat{w}$ which minimizes the following objective function:

$$\hat{w} = \arg\min_{w} \left[\frac{1}{n} \sum_{i=1}^{n} L(w^T X_i, Y_i) + \lambda w^T w, \right]$$

where $\lambda \geq 0$ is an appropriately chosen regularization parameter, which is used to stabilize the solution, and

$$L(p, y) = \begin{cases} \max(0, 1 - py)^2 & \text{if } py \geq -1 \\ -4py & \text{otherwise} \end{cases} .$$

It can be shown (Zhang, 2004) that by using this method, we have a probability model

$$P(Y = 1|X) = \max(0, \min(1, (1 + \hat{w}^T X)/2)). \qquad (6)$$

This probability estimate can then be combined with (5) to obtain a retrieval strategy, in which we retrieve every document X such that $\hat{w}^T X \geq -2u_{nr}/(u_r - u_{nr}) - 1$.

Another more classical linear model for probability estimation is logistic regression. Since threshold determination is particularly important to the categorization task (the performance is very sensitive to slight mistakes in the threshold value), we experimented with a few methods for threshold determination, including probability estimation and direct cross-validation. In addition, we experimented with feature weighting schemes such as binary versus TFIDF, as well as the ASO semi-supervised learning method

aibmadz05s	ASO w/ partially-supervised auxiliary problems; cross-validation threshold.
aibmadz05m1,2	aibmadz05s + supervised w/ binary features.
eibmadz05s	supervised w/ binary features; probability threshold.
eibmadz05m1,2	eibmadz05s + supervised w/o bi-gram features.
gibmadz05s	ASO w/ unsupervised auxiliary problems; probability threshold.
gibmadz05m1,2	gibmadz05s + supervised
tibmadz05s	ASO w/ partially-supervised auxiliary problems; probability threshold.
tibmadz05m1,2	tibmadz05s + supervised configuration w/ binary features.

Figure 5: Descriptions of categorization runs. m1 and m2 determine thresholds by probability estimation and cross validation, respectively. Auxiliary problems for ASO are described in (Ando and Zhang, 2005a).

briefly described in Section 2.1. In this study, we did not attempt to explore complicated document representation features, except for the use of Mesh. Our interests focus on the application of semi-supervised learning in this scenario, as well as threshold determination. The features we used are: tokens extracted from the title, abstract, and body sections of the journal articles; and the keywords from the Mesh sections of the corresponding Medline entries. In addition, we use the bi-grams that combine each of these tokens and the presence/absence of "Mouse" in the Mesh sections. No external knowledge source other than Medline is used.

Although ASO has been shown to effectively exploit unlabeled data on text categorization, unfortunately, this particular task is not ideal for experimenting with semi-supervised learning. One issue we encountered here is the availability of unlabeled data, which should ideally be taken from the same source as the test data. However, the sources of test data are biomedical journals, to which we did not have access for copyright reasons. As a substitute, we only used as unlabeled data a set of Medline abstracts (which are substantially shorter than full journal articles) that contain "mouse", "mice", or "mus". Another issue is the absence of relevant features. The supervised performance seems to indicate that our feature space does not contain strongly discriminating features.[6] In this situation, the performance bottleneck is the absence of relevant features from the designed feature space rather than the paucity of labeled training examples, and therefore, it becomes harder to benefit from unlabeled data. The third issue is the peculiarities of the evaluation metric used in this task: the performance is dominated by the proper determination of threshold values. In fact, even with the first two issues mentioned above, ASO was able to produce appreciable improvements over the baseline with oracle thresholds that are optimally selected on the test data. However, the performance gain becomes insignificant when thresholds are estimated on the training data, which implies that a good threshold estimation method is the key to the success on this task.

Due to the importance of threshold estimation, in addition to the simple method based on (5) and (6), we also considered a few more complicated ideas that can potentially improve the simple probability estimates. The three runs submitted for each sub-task were generated as follows. We performed cross validation of various configurations on the training data. We selected and combined the two best-performing configurations by fitting their outputs on the held-out data (part of the training data) using logistic regression. We applied two types of threshold determination methods (described below) to the combined classifier, which made two runs. The third run simply used the best-performing configuration.

One threshold determination method employed is 5-fold cross validation. The other is based on probability estimation based on (6) for modified Huber loss, and we use the standard probability estimate $1/(1+\exp(-\hat{w}^T X))$ if $\hat{w}$ is trained using logistic regression. With $u_{nr} = -1$, we can then threshold the estimated

[6] As mentioned in the report on the 2004 participation (Dayanik et al., 2004), we conjecture that there are some factors other than the content of the articles that are affecting the label assignments.

	x=a	x=e	x=g	x=t
$x$ibmadz05m1	.8492	.8277	.4993	.8931
$x$ibmadz05m2	.8482	.8339	.5004	.8998
$x$ibmadz05s	.8710	.8464	.4717	.8944
median	.7785	.6548	.4575	.7610
best	.8710	.8711	.5870	.9433

Figure 6: Categorization utility results.

probability according to (5) at $1/(u_r + 1)$.

3.3 Results

Figure 6 shows the utility results of our official runs (described in Figure 5) in comparison with the best and median utility among all the participants. It shows that our submission systems are competitive. All of our 12 official results are higher than the median. In particular, our aibmadz05s (ASO) achieved the best result among all the participants on sub-task A.

One issue with this task is that the results are quite sensitive to the threshold. Therefore we focused our efforts on reliable estimation of optimal thresholds. We did not spend much time on engineering features that explore domain specific knowledge, which might also significantly improve the overall system performance. As we mentioned earlier, semi-supervised learning is not necessarily suitable for this task, due to the particulars of the task setting.

4 Conclusion

Though our participation was in the Genomics Track, we focused on generally applicable approaches. We considered the ad-hoc retrieval task from the viewpoint of machine learning, and on the categorization task, we experimented with a regularized linear classifier in supervised and semi-supervised settings.

A central theme of our study is to explore the use of unlabeled data, both in information retrieval and in text categorization. We showed that the newly proposed structural feedback method, based on ideas from semi-supervised learning, consistently improves retrieval performance.

For the purpose of achieving good performance specifically on the Genomics track, a weakness of our approach is that we did not explore domain knowledge effectively. For the ad-hoc retrieval task, we did not include any system components to handle topic templates, which were designed for this year's Genomics track. In the categorization task, no domain specific information other than Medline Mesh terms was used. Although adding some domain specific components may potentially lead to better results on the two tasks in this year's Genomics track, it is encouraging to see that good performance can already be obtained from the general purpose techniques we investigated here.

Acknowledgments

We would like to thank James Cooper and Hironori Takeuchi for providing us with the synonym data extracted from domain databases. We would also like to thank Anni Coden and David Johnson for helpful discussions.

References

Rie Kubota Ando and Tong Zhang. 2005a. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Rie Kubota Ando and Tong Zhang. 2005b. High-performance semi-supervised learning method for text chunking. In *Proceedings of ACL-2005*.

A. Dayanik, D. Fradkin, A. Genkin, P. Kantor, and D. Mardigan. 2004. DIMACS at the TREC 2004 Genomics Track. In *Proceedings of the Thirteenth Text Retrieval Conference TREC 2004*.

S. Fujita. 2004. Revisiting again document length hypotheses TREC 2004 Genomics Track experiments at Patolis. In *Proceedings of the Thirteenth Text Retrieval Conference TREC 2004*.

S.E. Robertson, S. Walter, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. 1994. Okapi at TREC-3. In *Proceedings of the Third Text Retrieval Conference (TREC-3)*.

A. Schwartz and M. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proceedings of the Pacific Symposium on Biocomputing (PSB 2003)*.

Tong Zhang. 2004. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–85. with discussion.