BioText Team Report for the TREC 2003 Genomics Track

G Bhalotia[†], PI Nakov[†], AS Schwartz[†], MA Hearst[§]
[†]Computer Science Division
[§]School of Information Management and Systems
University of California Berkeley
Berkeley, CA 94720
{bhalotia, nakov, sariel}@cs.berkeley.edu, hearst@sims.berkeley.edu

Abstract

The BioText project team participated in both tasks of the TREC 2003 genomics track. Key to our approach in the primary task was the use of an organism-name recognition module, a module for recognizing gene name variants, and MeSH descriptors. Text classification improved the results slightly. In the secondary task, the key insight was casting it as a classification problem of choosing between the title and the last sentence of the abstract, although MeSH descriptors helped somewhat in this task as well. These approaches yielded results within the top three groups in both tasks.

1 Introduction

The paper reports on the work conducted by the BioText project team at UC Berkeley for the TREC 2003 Genomics track. In 2003 this track consists of two tasks and the document collection consists of 525,938 MEDLINE records dating between 4/1/2002 and 4/1/2003. Task 1 was intended to be similar to standard information retrieval queries and was stated as follows:

> For gene *X*, find all MEDLINE references that focus on the basic biology of the gene or its protein products from the designated organism. Basic biology includes isolation, structure, genetics and function of genes/proteins in normal and disease states.

The relevance judgements for this task were drawn from GeneRIF references from the National Library of Medicine's LocusLink database. Participants were allowed to make use of the gene name variation information associated with the GeneRIF.

The secondary task was intended to require more detailed analysis, in order to allow groups to make use of sophisticated language processing technology that is generally considered to be important for genomics and other bioscience text. However, due to limited resources, a specialized annotated collection was not yet available for this task. Instead, the goal of the secondary task was to reproduce the GeneRIF textual description for a given gene/document pair. Because these descriptions were often extracted verbatim from the document's title or abstract, and systems were judged on how closely the extracted text overlapped with the original, the task was better approached as a classification problem than as a language analysis and generation problem.

There were some commonalities in our approaches for the two tasks: in both cases we made use of classification algorithms and a special module for recognizing gene name variants and identifying MeSH descriptors in the text. Below approaches to both tasks are described in detail.

2 The Primary Task

2.1 Overview

The main challenges in the primary task are to improve recall by finding all appropriate variations of the given gene name, to improve precision by removing documents that describe genes that do not pertain to the target organism, and to demote the ranking of documents that mention the target gene but have not been assigned to a GeneRIF.

To improve recall, we created a special-purpose algorithm for generating and recognizing gene name variants. Our three-fold approach consisted of normalizing gene names by replacing special characters with spaces, developing a set of expansion rules that

generate possible variants of the gene names that are not included in LocusLink, and looking in the citations for MeSH terms that pertain to gene names.

To improve precision we developed a semi-automated method to convert LocusLink organism names to MeSH organism descriptors, and used these to filter out papers that were not relevant to the target organism.

We submitted two runs, illustrated in Figure 1. For the first run, the relevance ranking component consisted of a weighted sum over 5 different sub-queries. For the second run, this score was combined with that of a statistical model that was trained to distinguish documents that are referred to by GeneRIFs from those that are not. We used as our backend retrieval system the IBM DB2 Net Search Extender, which allows convenient combination of relational and full-text queries.

In the following subsections we describe the modules for gene name recognition, organism filtering, MeSH term mapping, GeneRIF classification, and the method of combining the scores of the sub-queries.

2.2 MeSH Descriptors

In a number of places in the paper below we make use of MeSH (Medical Subject Heading)[1] lexical hierarchy.

In MeSH, each concept is assigned a unique identifier (e.g., *Eye* is D005123) and one or more alphanumeric tree numbers (corresponding to particular positions in the hierarchy). For example, A (*Anatomy*), A01 (*Body Regions*), A01.456 (*Head*), A01.456.505 (Face), A01.456.505.420 (*Eye*). Eye is ambiguous according to MeSH and has a second tree number: A09.371 (A09 represents *Sense Organs*).

In addition, each MeSH concept is assigned a semantic type (there are over 200): e.g., *Enzyme*, *Gene or Genome*, *Mammal*, *Tissue*, *Virus*, etc.

In some cases in the work below we use MeSH tree numbers and truncate them at period breaks to generalize across sub-hierarchies of the trees. In other cases we use the unique descriptor or the semantic type.

2.3 Identifying Variations in Gene Names

In order to capture the variations in gene names we had to expand the original synonym list from LocusLink since using only the gene synonyms available from LocusLink produced relatively less accurate results.

[1] http://www.nlm.nih.gov/mesh

We created a semi-automated technique to identify such variations. We analyzed a large set of gene names to try to determine rules for converting a given representation into a canonical form. First we used n-gram matching to find candidate sets of similar gene names. Then we inspected the results of this matching to make a set of rules for such conversion (see Table 2). Some of these rules were more accurate and so were assigned more weight than the others. The details appear in the next two subsections.

N-Gram Overlap

The first step in identifying patterns of variation is to locate the variant form of the gene name in the article text. We automated this step by using an n-gram overlap measure [3]. "n-grams" are simply n-long strings of continuous characters in a given document/string. The distribution of n-grams between pairs of strings is compared, and a score is computed that represents the similarity between them. The main idea behind using n-grams is that similar words will have a high proportion of n-grams in common. Typical values for n are 2 or 3 corresponding to the use of digrams or trigrams, respectively.

To compute the similarity of two strings using this method, we first compute the n-gram sets for the strings being compared and then calculate the overlap using the Dice Coefficient [10]. The Dice coefficient $\mathcal{D}$ for two sets A and B of sizes $|A|$ and $|B|$ is given by

$$\mathcal{D} = \frac{2|A \cap B|}{|A| + |B|}$$

This overlap measure penalizes the presence of extra characters beyond the ones common to the two strings. Thus two strings with the same amount of overlap get a higher score when the non-overlapping regions are smaller in size.

We take the abstract and title of the articles associated with the genes and compute the n-gram overlap of all the possible subsequences of words against all known alias forms of the gene. We use character level digrams and trigrams with Dice Coefficient as the overlap measure. The word sequences in the abstracts/title that have high similarity to one of the known alias forms of the query gene are reported.

Inspection and Rule Generation

The procedure outlined above yields high similarity pairs of strings with one of them corresponding to the known alias form of a gene name and the other to the actual variant form of representation found in article text. We process this list to remove the

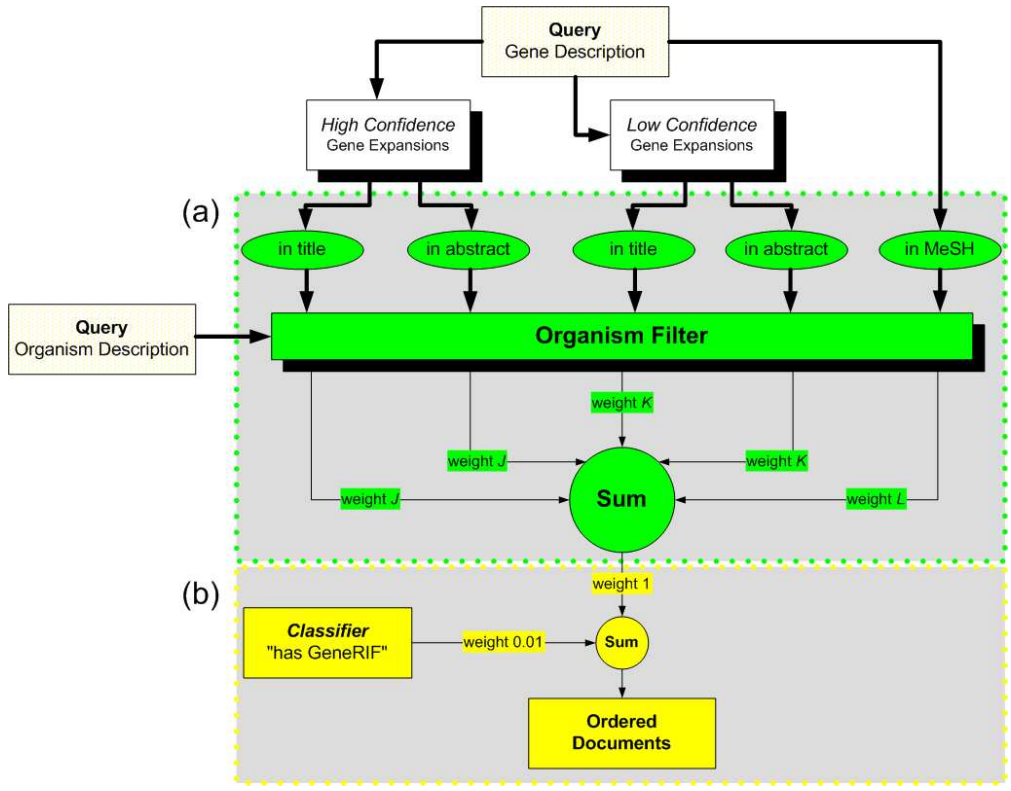

Figure 1: Software architecture for the primary task. Part (a) was used for both runs; part (b) for the second run only.

Known Alias Name	**Best match variant in text**
HLA-DQB1	hla-dqb
DNA synthesis inhibitor	inhibitors of dna synthesis
phospholipase C, gamma 1	phospholipase c gamma 1
adrenergic receptor, alpha 1d	alpha 1d-adrenergic receptor
Janus kinase 2 (a protein tyrosine kinase)	protein tyrosine kynase
golgi protein, 73-kD	golgi protein
luteinizing hormone/choriogonadotropin	luteinizing hormone–choriogonadotropin (lh/hcg)

Table 1: Some selected overlap pairs for gene names and their variants

ones that are exactly identical (note that identical strings will receive the highest similarity coefficient of 1). Next we remove those that lie below a threshold, that is obtained using quick inspection of the list (we used a threshold of 0.5). This yields a set of original forms and their variants. Selected overlap matches are shown in Table 1.

We inspect the pairs obtained to identify the patterns of variation in gene names. These patterns are used to generate rules to transform the names to obtain a broader set of alias forms for the gene names. The rules that we generated are shown in Table 2. Such rules are syntactic in nature; sometimes there are variations of semantic nature that cannot be captured this way.

2.4 Organism Filtering

We filtered out documents that do not correspond to the organism that the query gene belongs to (note that each query in the TREC task consists of a gene name and a corresponding organism). Similar genes with the same name can occur in multiple organisms, e.g., the gene named c-myc which stands for "cellular myelocytomatosis oncogene" can be found in different organisms including humans and chickens. In humans it is located on chromosome 8 and is involved in the pathogenesis of Burkitt's lymphoma. In chickens, c-myc activation by avian leukosis virus appears to result in the development of lymphoid leukosis. Most of the time we are interested in documents that talk about the function of the gene corresponding to a given organism.

MEDLINE records do not contain an "organism annotation" for the documents. However this information can be inferred from the MeSH terms assigned to the article. For example, a document that talks about the fruitfly "Drosophila melanogaster" contains the MeSH term *Drosophila melanogaster*. However the organism names used in LocusLink usually do not match the corresponding MeSH term. For example, the term *Human* is used in MeSH instead of "Homo sapiens" used by LocusLink.

We used the combined information in LocusLink and MEDLINE to identify the descriptors used to characterize the organisms for MEDLINE documents. We collected the MEDLINE references (as described before, LocusLink has a set of references to MEDLINE documents relevant to the gene) for documents corresponding to each organism in LocusLink.[2] Each query produced a set of documents corresponding to a LocusLink organism. We then ran a query to compute what the top MeSH terms were for each set of documents.

By looking at the most frequent MeSH descriptors for each of the document classes, we can infer the term that is used to denote the organism in MEDLINE. We also checked if the LocusLink organism is used in MEDLINE name in full or partially for the same one; e.g., the terms *Drosophila* and *Caenorhabditis* also appear in the MeSH headings contained in MEDLINE. None of the other organism names appears in their original LocusLink form. The terms that we ultimately use to map the documents in the collection to organisms are shown in Table 3. Some of the documents map to multiple organisms and a few map to none.

2.5 Mapping Query Terms to MeSH

In order to further improve the system's performance, we also retrieved documents using their MeSH annotations. Both MeSH Main Headings and MeSH Supplementary Concepts (Chemical List) map to MeSH concepts. Each MeSH concept has one or more textual synonyms that are called MeSH terms. Given a query term the system retrieves all the documents that are annotated with a MeSH concept that has a MeSH term that exactly matches one of the gene names or one of its original synonyms (not including the expanded forms).

Adding MeSH mappings to the query helped mainly in ranking the retrieved documents. Documents that are retrieved by using the MeSH mapping in addition to the text search are more likely to be relevant, and therefore are given higher scores.

2.6 GeneRIF Classification Module

The goal of the classification module is to estimate the probability that a given document has been assigned to a GeneRIF. Our approach is based on the idea that articles which discuss gene function contain a distinct set of features which can be learned using automated techniques. The resulting models can be used to classify new documents.

Feature Selection

We experimented with a number of different feature sets; a comparison of their corresponding classification accuracy is presented in Figure 2. We compared (a) using MeSH descriptors as complete phrases, (b) using MeSH tree numbers, (c) using words in abstracts with stemming, and (d) using MeSH descriptors combined with tree numbers from levels one and two. The best results are obtained for MeSH descriptors used as complete phrases.

[2]LocusLink contains genes from eight different organisms.

A, B	$\Rightarrow$	$\begin{cases} A\ B & \textit{removal of comma} \\ B\ A & \textit{rearrangement of tokens} \end{cases}$
$\left.\begin{array}{l} A([n]) \\ A_[n] \\ A\text{-}[n] \end{array}\right\}$	$\Rightarrow$	$A\ [n]$ *where* $[n]$ *is the set of numerals*
$A[n]$	$\Rightarrow$	$A\ [n]$ *addition of spaces (normalization of numerals)*
$A\ [n]$	$\Rightarrow$	$A[n]$ *removal of spaces (denormalization of numerals)*
$A(B)$	$\Rightarrow$	$\begin{cases} A\ B & \textit{removal of parentheses} \\ A & \textit{removal of terms in parentheses} \end{cases}$

Table 2: Rules for expanding gene names

Organism name from LocusLink	MeSH descriptor to look for
Bos taurus	Cattle
Caenorhabditis elegans	Caenorhabditis elegans, Caenorhabditis
Danio rerio	Zebrafish
Drosophila melanogaster	Drosophila melanogaster, Drosophila
Homo sapiens	Human
Human immunodeficiency virus 1	HIV-I
Mus musculus	Mice
Rattus norvegicus	Rats

Table 3: MeSH terms used to map documents to organisms

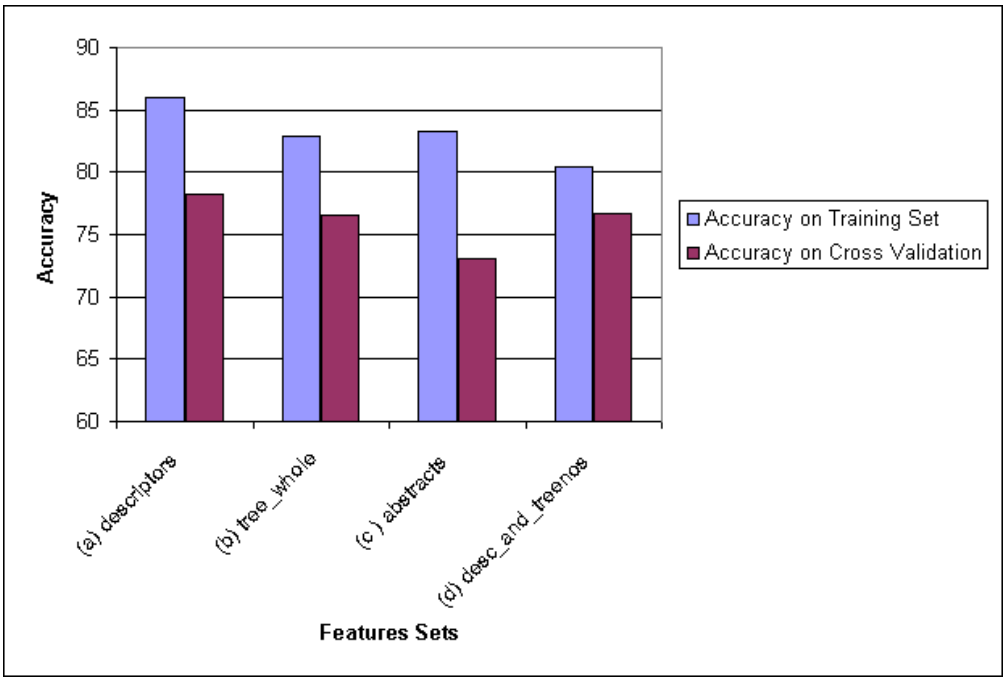

Figure 2: Classification accuracy for different feature sets: (a)descriptors as phrases (b) Whole tree numbers (c) Abstracts cleaned and stemmed (d) descriptors together with tree numbers

Model Training

Once the feature vectors are obtained, we train a classifier to build a model that can predict whether a document talks about gene function. We use a Naive Bayes classifier [5, 9]. Its fundamental idea is the assumption that the values of the feature variables $F = (F_1, F_2, ..., F_n)$ are conditionally independent given the class variable S. The joint probability is given by the expression:

$$p(S, F) = p(S) \prod_{i=1}^{N} p(F_i|S) \quad (1)$$

The model parameters are given by the probabilities $p(S)$ and $p(F_i|S)$, which are usually estimated from the text by means of maximum likelihood estimates (MLE). The classification of a new concept is determined by the most likely category:

$$S_{ML} = \arg\max_{S_k} p(S_k|F) \quad (2)$$

Naive Bayes classifiers are among the most successful algorithms for document classification. The Naive Bayes classifier is known to be optimal when attributes are independent given the class, but Domingos et al. [4] show that it will often outperform more powerful classifiers for common training set sizes and numbers of attributes even if the independence assumption is not met.

The implementation of the Naive Bayes classifier we currently use is part of the open source machine learning package WEKA (Waikato Environment for Knowledge Analysis [1, 11]) from the University of Waikato, New Zealand. They provide excellent Java implementation of Naive Bayes and other machine learning algorithms.

One model was trained for a set of 50 gene names that are not part of the TREC training or test set. We used the retrieval module to extract the relevant documents for each target gene, and the first 1000 documents of each query as the training set.

2.7 Document Ranking

As mentioned above, we used IBM DB2 Universal Database to store MEDLINE documents including the abstracts, titles and other annotations. We have built text indexes on these fields using DB2 Net Search Extender, which is then used to search for documents that contain a given set of terms.

We retrieve all the documents that match one of the known alias forms of the gene or the variations created using the expansion rules (variant forms are generated for all the aliases). The query against the database is composed of five sub-queries combined with the SQL UNION operator, as follows.

Let G be the various forms of the gene name as computed by the conversion rules shown in Table 2 and let LG be other lower-confidence rules for normalizing the gene names that have a higher rate of false positives. For the first run, the score is computed as follows:
Score(R) = the aggregated SUM over the result of the UNION operator GROUP BY document id of:

(a) J * (G compared to terms in titles)
(b) J * (LG compared to terms in abstracts)
(c) K * (LG compared to terms in titles)
(d) K * (LG compared to terms in abstracts)
(e) L * (MeSH concepts compared to MeSH terms assigned to documents)

where $J = 1, K = 0.015$, and $L = 1.4$ (determined experimentally on training data).

As shown above, the scores of the documents in each sub-query are weighted and then aggregated using the UNION operator and the SUM aggregate function. We experimented with using the MAX aggregate function instead, but the results obtained using the SUM function were substantially better. This is due to the fact that documents that are retrieved in multiple sub-queries get a higher total score, and are in fact more likely to be more relevant to the query. We also experimented with giving higher weights to titles over abstracts, but this did not appear to help. Increasing the weights of specific types of aliases (official terms for example) did not improve the system's performance either.

For the second run, in order to combine the retrieval scores and the classification scores we normalized the weighted scores for each query into values between 0 and 1 by dividing the score by the highest document score of the query. The combined retrieval-classification score is a weighted sum of the two scores. We used 1 and 0.01 as the weights for the retrieval and the classification scores respectively.

The retrieval score is a value between 0 and 1 for each document and is obtained in part by combining the frequency of occurrence of the term in the document and the relative size of the retrieved document. The exact details of the scoring function are not available as they are part of the DB2 proprietary system.[3]

[3]However, we have been told via personal communication from James Cooper of IBM, reporting information from Roy Byrd of IBM, that the algorithm is based on the Guru ranking algorithm [8], which is a Bayesian computation of a document's probability of being relevant to the query, with lexical affinities mixed in.

2.8 Evaluation

We submitted two runs for the primary task. The first run uses only the retrieval module, and the second run combines it with the classification module.

When training our system we noticed that some of the topics in the training set did not have any GeneRIFs associated with them. We therefore removed them from the training topic list. Also, some correct GeneRIFs were not listed in the list of qrels. After fixing these errors our system achieved 0.5028 mean average precision (MAP) on the training set with the retrieval module alone, and 0.5101 with the modules combined. The classification module helps but not dramatically.

On the test set our system achieved 0.3753 MAP with the retrieval module alone, and 0.3912 MAP using both modules combined. Again, the classification module helps but not markedly. In 12 out of 50 queries the retrieval module alone achieves MAP higher than the combined modules.

The big gap in performance between the test and training sets suggests that the system parameters might be over-fitting the training set. However, an initial sensitivity analysis of the system performance on the test set, shows that only a minor improvement can be achieved by tuning the parameters to fit the test set. Another explanation for the performance gap might be that the test set is inherently harder than the training set. This hypothesis is in agreement with the analysis presented in [6], which shows a high degree of variation in MAP across topics in general, and between the training and test sets in particular.

Both our runs fell within statistical significance of the top performing group [6]. In 43 out of 50 queries the MAPs of both runs were higher then the median MAP. Analysis of the 7 remaining queries yields some interesting insights about possible improvements of our system. In 3 out of 7 queries the low MAP was a result of a low recall. This is mainly due to some limitations of the current gene name expansion rules. For example, in query 37 the system retrieved only 36 relevant documents out of the possible 61. This is due to the fact that one of the query terms was *peroxisome proliferative activated receptor gamma* while in many relevant documents it appears as *peroxisome proliferative activated receptor gamma isoform 1*, and another query term, *PPAR gamma*, appears in the text as *PPARgamma*. Additional low-confidence expansion rules could improved the MAP in these cases without affecting the performance of other queries. In many cases the system retrieved much less than the allowed 1000 documents. In these cases, recall could also be improved by retrieving additional documents with low ranking scores like the ones that were filtered out by the organism-filter, or documents that could be retrieved using single query terms instead of full phrases. Of the other 4 sub-par queries, 1 was due to a small bug in the implementation of one of the high-confidence expansion rules that resulted in the addition of an over-generalized term with a high weight into the query, and the other 3 were due to sub-optimal rankings.

3 The Secondary Task

3.1 Overview

For the secondary task, our initial intention was to try a linguistically motivated approach but we soon realized that the data was too noisy due to a lack of clear definition of what a GeneRIF is. Instead, we addressed it as a text classification problem.

Our investigations showed that most of the time the GeneRIF text was pulled verbatim, or with slight modifications, from the abstract text or title. Most of the time the extract came from the title and in the majority of the remaining cases – from the last sentence of the abstract. Thus we assumed the baseline was always choosing the title since it was quite difficult to find an algorithm that performed better than this.

After much experimentation, we ended up training a Naive Bayes classifier that, given an abstract text, predicts whether the last sentence or the title is a better candidate for GeneRIF text. Our feature set was limited to verbs, MeSH terms (cut at level 2, e.g. G14.330), genes (a single feature for the frequency of all genes), all weighted by TF.IDF, and the appearance of the target gene (a Boolean feature). For training we focused on the abstracts coming from the 5 target journals that were announced on the genomics track Web site (about 6,500 abstracts in total), split them into 10 sets and performed a stratified 10-fold cross validation.

3.2 GeneRIF Mapping into the Abstract Text

Looking at the GeneRIFs we found that most of the time the GeneRIF text was pulled verbatim, or with slight modifications, from the abstract text. To quantify this, we investigated 33,662 MEDLINE abstracts that had a GeneRIF assigned and we tried to find a substring in the text that is most similar to the GeneRIF description. Given a particular abstract, we considered all possible sequences, respecting the

Run	Modified Unigram Dice
In title	35.17%
In abstract	45.81%
In both	4.98%
In last sentence	20.63%
Exact title match	19.67%
Total matched	76.02%

Table 4: Finding the best mapping of the GeneRIF text against the corresponding abstract.

word and sentences boundaries, and for each one we calculated the *Modified Unigram Dice* (*MUD*) score. We accepted a mapping as successful if the score was above some threshold.

When the MUD threshold was set to 80%, a successful mapping for 25,590 of the documents (76.02%) could be obtained directly from the title and/or abstract. For 11,847 (35.17%) of them an acceptable match was a substring of the title (and for 6,620 (19.67%), the whole title was taken verbatim: this is 65.10% of the cases when the mapping was found in the title). In 15,421 documents (45.81%), the best match was inside a sentence from the abstract body, and in 1,678 (4.98%) it was found in both the title and the body. In 6,943 of the cases (20.63%) the best match was found in the last sentence of the abstract (this is 50.52% of the cases when it was found in the abstract body). (Note that there is always further opportunity to truncate some unused part of the sentence and improve the score.)

Given the fact that for most of the abstracts an acceptable matching was found in the title and that in 65.10% of them the best match was the whole title taken verbatim, an obvious baseline was "pick the title". This resulted in a MUD score of 53.39%.

The last sentence of an abstract usually summarizes its contents, so it was not surprising that it often contained the best match. The title and the last sentence together account for 73.40% of the matches that pass the threshold. Thus, an algorithm that chooses between them would have the potential to perform better than always choosing the title. We calculated that if we limited the choice to title or last sentence and always selected the one that leads to a higher score (we select the *whole* last sentence or the *whole* title), this would result in a MUD score of 66.33%. This is the upper bound for any algorithm that relies on whole sentences and chooses between the title and the last abstract sentence only. In practice this algorithm may not perform better than the "pick the title" baseline since it may choose incorrectly. We calculated that if it always made the wrong choice it would end up with a MUD score of 26.62%.

We performed similar calculations using as similarity measures *Classic Dice* (*CD*), *Modified Bigram Dice* (*MBD*), *Modified Bigram Dice Phrases* (*MBDP*) as well as a combination of MBU and MBD. The results were all similar, although sometimes the best choices under the different scores differed. We also performed a more general mapping that allowed the target GeneRIF text description to split into two parts, each of which can be mapped to two different parts of the abstract text. Although this way we found better matches for some of the GeneRIFs, the impact was limited: 77.10% matched as compared to 76.02% for the case when a single string was allowed.

3.3 The Features

We experimented with a number of different features, including: *words/stems*, *verbs* (the most frequent ones only: e.g. *bind*, *block*, *inhibit*, *accept*, *involve* etc.; they are stemmed so nominalized verbs are considered as well: e.g. *inhibition*), *genes*, *genes_freq* (frequency: how many gene names are mentioned in the sentence), *MeSH_unique_ID* (e.g. *D005796*), *MeSH_tree_number* cut at a certain level (level 1: *G14*, or level 2: *G14.330*), *MeSH_semantic_type*, *journal*, *publication_date* (month and year taken together, e.g. *10_2003*). We used also three Boolean features: *target_gene* (is the target gene mentioned?), *is_title* (is the current sentence the title?), *is_last_sentence* (is this the last sentence?). The features were weighted according to the TF.IDF measure (except for the Boolean ones). We also experimented without weighting (i.e. using the raw frequency information) as well as treating all features as Boolean.

The *journal* and *publication_date* features were introduced in order to account for possible journal- or time-dependent regularities, but these did not prove useful. The *MeSH_semantic_type* features were too general. The *words* and *stems* lead to a dramatic increase in the vector space dimensionality, which made training some particular classifiers intractable so we did not use them. The same applies to *genes_freq* and *MeSH_unique_ID*.

The best combination of features was (we will refer to it as *the standard feature set* below): *verbs*, *genes_freq*, *MeSH_tree_number* (cut at level 2), *target_gene*, *is_title* and *is_last_sentence*. The three Boolean features were especially important, while the impact of *genes_freq* was minor. The non-Boolean features were weighted using TF.IDF.

3.4 Choosing the Title vs. the Last Sentence

For the classification experiments we used the WEKA Machine Learning Software in Java again. We used mainly Naive Bayes with kernels [7] but tried several others classifier as well. Decision trees were helpful to identify the useful features: e.g. the MeSH terms turned up often.

As mentioned above, we addressed the problem as a text classification task at the sentence level, and ended up stating it as a choice between the title and the last sentence. Using the *standard feature set*, we trained a Naive Bayes classifier that was able to distinguish between when the best sentence is a title vs. a non-title with an accuracy of 81.22%, as measured on a stratified 10-fold cross-validation on a corpus of 4,000 GeneRIF texts.

We wanted to extend this idea in the direction of a two-step classification: the first classifier chooses between title and non-title. In case non-title is chosen, then a second classifier chooses the best abstract sentence (here the most important feature is *is_last_sentence*). The second classifier is to be trained on non-title sentences only. Unfortunately, comparing the title to the abstract body was problematic due to substantial length differences. We decided to simplify the things further and compare the title to the last sentence only.

We trained a Naive Bayes classifier with kernels that, given a gene and a document, chooses between the title (*class A*) and the last sentence (*class B*). We used the *standard feature set*, but without the *is_title* and *is_last_sentence* features. In order to label the training examples as belonging to class A/B we compared the MUD overlap of the target GeneRIF text with both the title and the last sentence and assigned the label A or B depending on which get a higher score. We then concatenated the title and the last sentence and extracted the features from the resulting string. So, each abstract produced a single example labeled either A or B.

We tried marking the features (e.g. MeSH terms) to indicate whether they came from the title or from the last sentence in order to allow the classifier to distinguish between them, but this lead to decreased performance and so was dropped. Finally, we limited the training to the abstracts coming from the 5 target journals that were announced on the genomics track Web site: about 6,500 abstracts in total. We split them into 10 sets and performed a stratified 10-fold cross-validation.

3.5 Evaluation

We performed several experiments for the different formulations of the problem in order to find the best feature set and the best classification algorithm using stratified 10-fold cross-validation and collections of different sizes: 343, 1000, 2000, 5000, 10000, 20000, 33662 etc. Then we fixed the feature set (the *standard* feature set) and the classifier (Naive Bayes; class A/B) and concentrated on a set of 6,500 abstracts that have GeneRIFs and come from the 5 journals. We ran series of experiments in order to find the best thresholds for feature selection.

The baseline results of always choosing the title for the various scoring metrics are shown on line 1 of Table 5. For the training set cross validation runs, the best results were obtained for minimum verb frequency of 5 and minimum MeSH tree number frequency of 12 (see line 2 of Table 5).

For the TREC run we trained on all the abstracts from the 5 journals, except the 139 ones used for testing. We used the feature thresholds found above (5 for verbs and 12 for MeSH tree numbers) and we obtained the results shown on line 3 of Table 5. Although these are lower than those of line 2, they are still about 3–4% above the baseline shown on line 1.

We also calculated the best possible score we could have obtained if our algorithm had always made the correct choice between the title and the last sentence (see line 5 of Table 5). If we had tuned the parameters to the test set, we would have used minimum verb frequency of 12 and minimum MeSH frequency of 11 and gotten the results shown in line 4 of Table 5. (Of course, we cannot use the test set to compute the thresholds for the TREC run.)

Finally, our scores 57.83%, 59.63%, 46.75%, 49.11% (for CD, MUD, MBD and MBDP) were the second best ones after those of Erasmus University (emc4): 53.04%, 54.65%, 38.62%, 41.17, who used a similar classification technique but managed to successfully train a classifier to choose among *all* sentences, not just between the title and the abstract. In fact, emc4 was the only other group that was able to beat the baseline. However, according to [6], neither of these results represents a statistically significant improvement over just using the titles.

4 Discussion

4.1 Primary Task

It appears that the definition of GeneRIF is quite fuzzy. This limits the potential contribution of our classification module. However, we believe that in

	Run	CD	MUD	MBD	MBDP
1	Baseline	50.47%	52.60%	34.82%	37.91%
2	Cross Validation	58.06%	59.11%	44.74%	47.29%
3	TREC run	53.04%	54.65%	38.62%	41.17%
4	Tuned thresholds	54.88%	56.66%	40.66%	43.31%
5	Upper bound	61.72%	64.19%	50.88%	54.00%

Table 5: TREC run result compared to baseline, best possible result and a better features selection.

cases where the subset of relevant documents could be defined more precisely combining such a classifier with a more traditional IR system could result in a significant boost in performance.

In order to further improve the performance of our system, semantic information has to be incorporated. In the future we plan to add syntactic and semantic annotations to the text in order to support much more powerful algorithms for information retrieval and extraction from bioscience literature.

4.2 Secondary Task

Better results could potentially be obtained with a careful feature selection algorithm [12]: all we do at present is to remove the least frequent features and our algorithm is very sensitive to setting the correct threshold (compare lines 3 and 4 in Table 5). This would allow the introduction of some carefully selected stems as features without a dramatic increase of the vector space dimensionality, and can account for predictive phrases of the kind: *our results show that* .... In addition, although good, our gene and MeSH tagging utilities are not perfect. MeSH ambiguity is another source of problems and the verb nominalization introduces some noise as well.

A promising improvement would be a more careful truncation of the unnecessary part of the selected sentence. It would be also interesting to try some specialized algorithm that tries to learn a ranking directly (e.g. [2]) as opposed to the classification approach described above.

Finally, it would be good to have a similarity measure that takes into account the semantics, e.g., not penalize those cases in which a synonym name for the same gene is used.

References

[1] Waikato environment for knowledge engineering. http://www.cs.waikato.ac.nz/ml/weka/.

[2] Crammer, K., and Singer, Y. A new family of online algorithms for category ranking. In *25th annual international ACM SIGIR conference on Research and development in information retrieval* (2002), pp. 151–158.

[3] Damashek, M. Gauging similarity with n-grams: Language-independent categorization of text. In *Science* (1995), vol. 267, pp. 843 – 848.

[4] Domingos, P., and Pazzani, M. Beyond independence: Conditions for the optimality of the simple bayesian. In *International Conference on Machine Learning, ICML* (1996).

[5] Duda, R., and Hart, P. Pattern classification and scene analysis, 1973.

[6] Hersh, W., Bhupatiraju, R., Kraemer, D., Corley, S., Aronson, A., Mork, J., Hearst, M., Nakov, P., and Mitchell, J. Enhancing access to the bibliome: The trec genomics track, 2004. In preparation.

[7] John, G., and Langley, P. Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence* (1995), Morgan Kaufmann, pp. 338–345.

[8] Maarek, Y., and Smadja, F. Full text indexing based on lexical relations. In *Proceedings of the 12th International ACM SIGIR Conference on Research and Development in Information Retrieval* (1989), pp. 198–206.

[9] Mitchell, T. *Machine Learning*. McGraw Hill, 1997.

[10] van Rijsbergen, C. *Information Retrieval*, 2nd ed. Butterworth-Heinemann, 1979.

[11] Witten, I., and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

[12] Yang, Y., and Pedersen, J. A comparative study on feature selection in text categorization. In *Proceedings of the Fourtheenth Intenational Conference on Machine Learning* (1997).