An Application of Text Categorization Methods to Gene Ontology Annotation

Kazuhiro Seki
Laboratory of Applied Informatics Research
Indiana University, Bloomington
1320 East Tenth Street, LI 011
Bloomington, Indiana 47405-3907
kseki@indiana.edu

Javed Mostafa
Laboratory of Applied Informatics Research
Indiana University, Bloomington
1320 East Tenth Street, LI 011
Bloomington, Indiana 47405-3907
jm@indiana.edu

ABSTRACT

This paper describes an application of IR and text categorization methods to a highly practical problem in biomedicine, specifically, Gene Ontology (GO) annotation. GO annotation is a major activity in most model organism database projects and annotates gene functions using a controlled vocabulary. As a first step toward automatic GO annotation, we aim to assign GO domain codes given a specific gene and an article in which the gene appears, which is one of the task challenges at the TREC 2004 Genomics Track. We approached the task with careful consideration of the specialized terminology and paid special attention to dealing with various forms of gene synonyms, so as to exhaustively locate the occurrences of the target gene. We extracted the words around the gene occurrences and used them to represent the gene for GO domain code annotation. As a classifier, we adopted a variant of k-Nearest Neighbor (*k*NN) with supervised term weighting schemes to improve the performance, making our method among the top-performing systems in the TREC official evaluation. Moreover, it is demonstrated that our proposed framework is successfully applied to another task of the Genomics Track, showing comparable results to the best performing system.

Categories and Subject Descriptors

H.2.4 [**Database management**]: Systems—*Textual databases*; H.3.1 [**Information storage and retrieval**]: Content Analysis and Indexing—*Abstracting methods, Indexing methods, Linguistic processing*; J.3 [**Life and medical sciences**]: Biology and genetics

General Terms

Algorithm, Performance, Experimentation

Keywords

Text Categorization, Automatic Database Curation, Genomic IR

SIGIR'05, August 15–19, 2005, Salvador, Brazil

1. INTRODUCTION

Given the intense interest and fast growing literature, biomedicine is an attractive domain for exploration of intelligent information processing techniques, such as information retrieval (IR), information extraction, and information visualization. As a result, it has been increasingly drawing much attention of researchers in IR and other related communities [6, 7, 9, 17]. As far as we know, however, there have been few products of research efforts focusing on this particular domain in past SIGIR conferences. This paper introduces a successful application of general IR and text categorization methods to this evolving field of research targeting biomedical texts.

In the post-genomic era, one of the major activities in molecular biology is to determine the precise functions of individual genes or gene products, which has been producing a large number of publications with the help of high throughput gene analysis. To structure the information related to gene functions scattered over the literature, a great deal of efforts has been made to annotate articles by using the Gene Ontology[1] (GO) terms. GO is a controlled vocabulary developed for describing functions of gene products in order to facilitate uniform queries across different model organism databases, such as FlyBase, Saccharomyces Genome Database (SGD), and the Mouse Genome Informatics (MGI) Database. GO terms are basically organized in hierarchical structures (but a child node may have multiple parent nodes) under three top level nodes: molecular function (MF), biological process (BP), and cellular component (CC). Figure 1 illustrates the structure of GO.

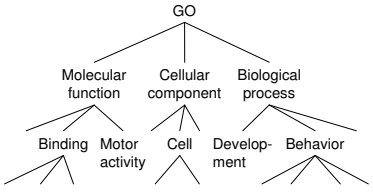

Figure 1: Structure of Gene Ontology.

Because of the large number of publications and specialized content, GO annotation requires extensive human efforts and substantial domain knowledge, which is usually conducted by experts. Thus, there is a potential need to automate or semi-automate GO annotation, which could greatly alleviate the human curation. This was one of the primary objectives pursued at the Text Retrieval Conference (TREC) 2004 Genomics Track [8].

[1] `http://www.geneontology.org/`

The 2004 Genomics Track consisted of two tasks: *ad hoc retrieval* and *categorization* tasks. For the former, given 50 topics obtained through interviews with real research scientists, the participants were required to find relevant documents from 10 years' worth of MEDLINE data. The latter task (which is our focus in this paper) was composed of two sub-tasks; one was called the *triage* task and the other the *annotation* task. Both tasks mimicked some parts of GO annotation process currently carried out by human experts at Mouse Genome Informatics (MGI). Figure 2 depicts the conceptual flow of the two sub-tasks.

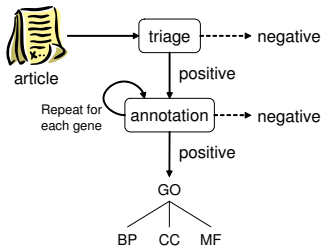

Figure 2: A conceptual flow of the categorization sub-tasks.

In short, the goal of the triage task was to correctly identify whether an input article contains experimental evidence that warrant GO annotation regardless of specific GO codes. The annotation task was the next step to the triage decision, and the goal was to correctly assign GO domain codes, i.e., MF, BP, and CC (not the actual GO terms) or not to assign them, i.e., negative, for each of the given genes that appear in the article.[2] Note that there may be more than one gene associated with an article and there may be more than one domain code assigned to a gene.

The triage task can be seen as a standard text categorization problem to classify an input article into predefined classes (positive and negative), while the annotation task required to classify not an article as a whole but each given gene appearing in the article. In other words, each ⟨article, gene⟩ pair was to be independently classified even when two (or more) genes appeared in a single article. We addressed the problem by extracting document fragments that were likely to contain the gene in question by gene name expansion and a flexible term matching scheme. The resulting set of document fragments were then used for representing the particular gene. For classification, we used a variant of the k-Nearest Neighbor (*k*NN) classifier with supervised term weighting schemes [2] which consider word distributions in different classes.

This paper focuses on an application of general IR and text categorization techniques to the domain-specific, highly practical problems with careful consideration of the properties of the terminology in biomedicine. In the following, we first introduce our proposed framework for GO domain code annotation, and then describe the data and evaluation measures used in our experiments. We show the effectiveness of our framework through a number of experiments with various different settings. In addition, it is demonstrated that our framework can be successfully applied to the triage task as well, making it among the top-performing systems for both triage and annotation tasks in the TREC official evaluation.

[2]Assuming perfect triage decision, there would not be negative cases at the annotation stage. However, there were negative instances purposefully included in the TREC data (Section 3.1).

2. METHODS

This section details our proposed framework for automatic GO domain code annotation. Hereafter, we will refer to GO domain code annotation as "GO annotation" for short.

2.1 Document representation

2.1.1 Identification of relevant paragraphs

GO annotation needs to be made not for each input article but for each gene for which there is experimental evidence that warrants GO annotation. Therefore, each ⟨article, gene⟩ pair needs to be treated as a "document" or "text" in the sense of text categorization. For this purpose, we propose a simple but effective approach to extract only the text fragments that are likely to contain the gene in question and treat a set of the extracted text fragments as a document associated with the ⟨article, gene⟩ pair. This process can be broken down into *gene name expansion* and *gene name identification*, each explained in the following.

Gene name expansion. Gene name expansion refers to a process to associate synonyms with a given gene name. Gene names are known to have several types of synonyms including aliases, abbreviations, and gene symbols. For instance, "membrane associated transporter protein" can be referred to as *underwhite*, *dominant brown*, *Matp*, *uw*, *Dbr*, *bls*, *Aim1*, etc. Therefore, all of these names should be searched to identify text fragments mentioning the gene. To obtain such synonyms, we used two sources of information: the article itself and a gene name dictionary. As described later in Section 3, the input article is annotated with SGML tags and there are two relevant fields, <KEYWORD> and <GLOSSARY>, in which a gene name and its synonym may be explicitly defined.[3] Incidentally, we also examined the use of body text because gene name abbreviations often appear with parentheses immediately following the official names [14]. However, it slightly degraded classification in our preliminary experiments and thus was not used in the following experiments.

As another source of gene name expansion, a gene name dictionary was automatically complied from existing databases. For this work, we experimentally used the SWISS-PROT [11] and LocusLink [12] databases. The resulting name dictionary contained 493,473 records, where each record had a gene/protein name as a keyword and lists its synonyms. Hereafter, we use the word "gene names" to refer to all of official names, aliases, abbreviations, and gene symbols.

It is often the case that gene name dictionaries automatically compiled from existing databases, such as ours, are not very accurate due to multi-sense gene names, inconsistent format in the databases, etc. It requires manual curation to obtain high-quality dictionaries [3, 5], which are important for general-purpose gene name recognition systems. Fortunately, the quality of dictionary would not be as important in our application, because even if the dictionary provides wrong gene names as synonyms of a given gene, the wrong names are unlikely to appear in the article as they are irrelevant to the target gene with which the article is associated.

Gene name identification. The next step is to find text fragments mentioning the gene in question. Here, the problem is that, besides synonyms, gene names often have many variants due to arbitrary use of special symbols, white space, and capital and small letters [3]. To tolerate these minor differences in identifying gene

[3]The DTD is found at `http://highwire.stanford.edu/about/dtd/`

names, both gene names and text were preprocessed as follows (the actual order is not important).

- Replace all special symbols (non-alphanumeric characters) with space (e.g., `NF-kappa` → `NF kappa`)
- Insert space between different character types, such as alphabets and numerals (e.g., `Diet1` → `Diet 1`)
- Insert space between Greek alphabets and other words (e.g., `kappaB` → `kappa B`)
- Lowercase all characters

Then, each paragraph (identified by SGML tags) in the article was scanned if it contained any of the gene names associated with the gene in question. Note that section titles were appended to each paragraph since they were often found to be descriptive. In addition, if the paragraph referred to figures and/or tables for the first time in the article, their captions were also appended to the paragraph.

We have so far obtained gene name synonyms and normalized both gene names and text to facilitate gene name identification. However, there remains another problem. That is, gene names are frequently written in slightly different forms with extra words, different word order, etc. For example, "*peroxisome proliferator activated receptor binding protein*" may be referred to as "*peroxisome proliferator activator receptor (PPAR)-binding protein*" where underlines indicate the differences. To deal with the problem, we used approximate word matching. To be precise, for each target gene name and each candidate which mentions any word composing the gene name, a word-overlap score defined below was computed.

$$Overlap(gene, candidate) = \frac{M - \alpha \cdot U}{N + \beta} \tag{1}$$

where M and U denote the numbers of matching and unmatching words, respectively; α is a penalty for unmatching words (set to 0.3); N is the number of words composing the gene name; and β penalizes shorter gene names (set to 2). If any candidate associated with a paragraph had a score exceeding a predefined threshold (set to 0.3), the paragraph was used to represent the ⟨article, gene⟩ pair after stopword removal based on the PubMed stopword list[4] and stemming by Lovins stemmer [10]. For instance, the example of "*peroxisome...*" above has five matching and two unmatching words, resulting in an overlap score of 0.55. Because it is greater than the threshold (0.3), the paragraph containing the candidate is extracted and used in part to represent the corresponding ⟨article, gene⟩ pair. Incidentally, the values of the parameters were determined based on our preliminary experiments on the training data.

Here, we treated a paragraph as a unit since it is thought to be organized in a single topic and seems to be an appropriate unit of extraction. In Section 4.2, we will examine other alternative units.

2.1.2 MeSH terms

Along with the article itself, we took advantage of external resources, specifically, Medical Subject Heading (MeSH)[5] terms assigned to the article. MeSH terms are controlled vocabularies developed at the national library of medicine (NLM) for indexing biomedical articles and are annotated by human experts at NLM.

For each input article, all the associated MeSH terms were obtained from the MEDLINE database[6] using Entrez Utilities.[7] Because these MeSH terms are annotated with articles (not with particular genes), they were added to *each* document (a set of paragraphs) representing a pair of the article and *any* gene coupled with it. Note that a special symbol `MESH+` was concatenated to each MeSH term so as to distinguish MeSH from other terms.

2.1.3 Feature selection

Feature selection identifies the features (terms) that are more informative in terms of classification according to some statistic measure, which not only reduces the size of data but often improves classification [19]. For this work, we applied the chi-square statistic method to the terms contained in the documents obtained through the previous steps.

Chi-square statistic of term t in class c is defined as:

$$\chi^2(t, c) = \frac{N(AD - CB)^2}{(A + C)(B + D)(A + B)(C + D)} \tag{2}$$

where c is one of the GO domain codes (BP, MF, and CC) or negative (NEG), A is the number of documents containing term t in class c, B is the number of documents containing t in classes other than c, C is the number of documents not containing t in c, D is the number of documents not containing t in classes other than c, and N is the total number of documents. For each term t, chi-square statistic was computed for every class, and the maximum score was taken as the chi-square statistic for term t; that is, $\chi^2(t) = \max_i \chi^2(t, c_i)$. Only the top n terms with higher chi-square statistics were used for the following processes. We empirically chose n=3000 based on our preliminary experiments, where 234 terms were highest with class BP, 173 with MF, 277 with CC, and 2316 with NEG.

2.1.4 Term weighting

Each ⟨article, gene⟩ pair was associated with a set of selected terms in the preceding steps. To apply kNN for classification as described in the next section, we converted it to a term vector adopting the classic vector space model [13] with conventional TFIDF (term frequency-inverse document frequency) defined as:

$$\mathrm{TFIDF}(t, d) = (1 + \log(\mathrm{TF}(t, d)) \cdot \log \frac{N}{\mathrm{DF}(t)} \tag{3}$$

where $\mathrm{TF}(t, d)$ is a term frequency of term t within document d, N is the total number of documents, and $\mathrm{DF}(t)$ is the number of documents in which term t appears. In cases where $\mathrm{TF}(t, d) = 0$ or $\mathrm{DF}(t) = 0$, $\mathrm{TFIDF}(t, d)$ is defined to be 0.

We also tested another term weighting scheme, so called *supervised term weighting*, proposed by Debole and Sebastiani [2]. It takes into account pre-labeled class information in training data and re-uses statistics computed in the feature selection step (e.g., chi-square statistics, information gain, ...) in place of IDF. We used TFCHI which is defined as a product of TF and chi-square statistics. Specifically, we tested two variants of the scheme, denoted as TFCHI_1 and TFCHI_2.

$$\begin{aligned} \mathrm{TFCHI}_1(t, d) &= (1 + \log(\mathrm{TF}(t, d)) \cdot \chi^2(t) \\ \mathrm{TFCHI}_2(t, d) &= (1 + \log(\mathrm{TF}(t, d)) \cdot \log(\chi^2(t)) \end{aligned} \tag{4}$$

2.2 kNN classifiers

We used a variant of kNN classifiers to assign GO domain codes to each pair of article and gene. kNN is an instance-based classifier

[4] `http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html`
[5] `http://www.nlm.nih.gov/mesh/`
[6] `http://www.ncbi.nlm.nih.gov/entrez/query.fcgi`
[7] `http://www.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html`

which is reported as one of the best classifiers for text categorization in both newswire and medical domains [18]. In brief, it classifies input v to one or more predefined classes depending on what classes its neighbors belong to. The decision rule can be expressed as:

$$\text{if } Score(c, v) = \sum_i sim(v, n_{c,i}) > t_c, \text{ then assign } c \text{ to } v \quad (5)$$

where n_c is the k nearest neighbors having class $c \in \{\texttt{BP}, \texttt{MF}, \texttt{CC}, \texttt{NEG}\}$, t_c is a per-class threshold, and $sim(v, n_{c,i})$ returns cosine similarity between the arguments. Threshold t_c can be optimized to maximize an arbitrary metric (e.g. F_1-score) using training data.

We slightly modified the scoring scheme to multiply the similarity scores by the number of k neighbors having class c, denoted as $|n_c|$.

$$\text{if } Score(c, v) = \sum_i sim(v, n_{c,i}) \times |n_c| > t_c, \text{ then assign } c \text{ to } v \quad (6)$$

It intended to boost the scores for more frequent classes within the k neighbors. This modification slightly but constantly improved classification (around 2% in F_1 score).

Although class c includes NEG, we did not apply the decision rule above for the negative class. In other words, if none of the GO domain codes was assigned to input, then it was considered to be negative. This ensures that an input does not have both positive (BP, MF, or CC) and negative classes together. It should be noted that, however, negative class does affect classification because more negative instances included in k neighbors generally lead to lower scores for the positive classes.

3. DATA AND EVALUATION MEASURES

3.1 Data sets

We used the same data set as the Genomics Track annotation task. The data set is composed of 504 full-text articles for training and 378 for test, both in SGML format. Each of the articles is associated with one or more genes and each gene is annotated with one or more classes (BP, MF, and CC) or negative by MGI curators. The total numbers of triplets ⟨article, gene, class⟩ are 1661 (589 positives and 1072 negatives) and 1077 (495 positives and 582 negatives) for the training and test data, respectively. Because gene names often contain Greek alphabets, character entities used for representing Greek alphabets (e.g., "`&agr;`" for α) were converted to the corresponding English spellings (e.g., `alpha`) in advance to facilitate gene name identification.

The training data were used for tuning several parameters including the number of k neighbors and per-class thresholds t_c in Equation (6) and were used as pre-labeled instances for kNN to classify the test data.

3.2 Measures

Following the TREC Genomics Track, we used micro-averaged F_1 score as an evaluation metric for GO annotation, so as to make our results comparable with the official evaluation. F_1 is defined as the harmonic mean of precision and recall as in Equation (7).

$$\begin{aligned} Recall &= \frac{\text{\# of classes correctly predicted by the system}}{\text{\# of pre-labeled classes}} \\ Precision &= \frac{\text{\# of classes correctly predicted by the system}}{\text{\# of classes predicted by the system}} \\ F_1 &= \frac{2 \times Precision \times Recall}{Precision + Recall} \end{aligned} \quad (7)$$

where classes are biological process (BP), cellular component (CC), and molecular function (MF) and do not include negative (NEG).

4. RESULTS AND DISCUSSIONS

4.1 Primary results

Our proposed framework for GO annotation was applied to the test data, where the per-class thresholds t_c in Equation (6) and other parameters including the number of k neighbors were optimized to maximize F_1 for each term weighting scheme using the training data. For instance, for TFCHI_2, k was set to 140 and t_c was set to 1276, 936, and 1790 for BP, CC, and MF, respectively. Table 1 compares our results (TFIDF, TFCHI_1, and TFCHI_2) on the test data and the representative results from the TREC official evaluation.

Table 1: The TREC official results and our results for GO domain code annotation (on the test data set).

		Prec	Recall	F_1
TREC	Best	0.441	0.769	0.561
	Worst	0.169	0.133	0.149
	Mean	0.360	0.581	0.382
Ours	TFIDF	0.549	0.642	0.592
	TFCHI_1	0.480	0.630	0.545
	TFCHI_2	0.508	0.731	0.600

Despite the simplicity of our method, it performed quite well, especially, TFIDF and TFCHI_2, as compared with the official results. In the following sections, we take a closer look at major features or components of our framework and empirically investigate their contribution.

4.2 Additional experiments

4.2.1 Alternative settings

We have made a number of arbitrary decisions in developing our framework. To investigate the effectiveness, we conducted several experiments with various different settings. Particularly, we were interested if the features or components below had made any impact.

- Gene name identification: We identified paragraphs that were likely to contain the target gene using approximate word matching (see Section 2.1). Did it actually improve GO annotation? To examine it, we used exact word matching to identify relevant paragraphs.
- Gene name dictionary: Assuming gene name identification above worked, did the dictionary for gene name expansion contribute to the performance? We tested our framework without the help of the dictionary.
- Glossary and keyword fields: Similarly, did the use of SGML tags, `<GLOSSARY>` and `<KEYWORD>`, for finding gene name synonyms improve GO annotation? We tested our framework without the information.
- MeSH terms: Did the inclusion of MeSH terms contribute to classification? We tested our framework without MeSH terms.
- Unit of extraction: Was a paragraph as a unit of extraction appropriate? We explored other units:

– Only the sentence containing the target gene (denoted as *G*)

– In addition to *G*, an immediately succeeding sentence (denoted as *G*+*S*)

– In addition to *G*+*S*, an immediately preceding sentence (denoted as *P*+*G*+*S*)

– The entire article irrespective of the target gene (denoted as *ART*)

Note that, however, we did not let both *G*+*S* and *P*+*G*+*S* exceed paragraph boundaries. Incidentally, our framework focusing on paragraphs would be placed between *P*+*G*+*S* and *ART*.

4.2.2 Empirical observations

Table 2 shows the best possible F_1 scores for each of the experimental settings above, where the training data were classified using leave-one-out cross-validation, while the test data were classified using the training data as before. Note that the bottom row "Default" used the same setting as TFCHI_2 in Table 1 but shows higher F_1 than TFCHI_2, because threshold t_c for *k*NN was optimized on the *test* data for the purpose of cross-setting comparison.

Table 2: Results for alternative settings. Numbers in parentheses under "Average" indicate percent increase/decrease relative to "Default".

Experimental settings		F_1		
		Training	Test	Average
Gene name identification	Exact match	0.354	0.411	0.383 (−34.8%)
Gene name dictionary	Unused	0.416	0.470	0.443 (−24.5%)
Glossary and keyword fields	Unused	0.543	0.635	0.589 (+0.3%)
MeSH terms	Unused	0.543	0.625	0.584 (−0.5%)
Unit of extraction	*G*	0.525	0.613	0.569 (−3.1%)
	G+*S*	0.532	0.619	0.576 (−2.0%)
	P+*G*+*S*	0.535	0.620	0.578 (−1.6%)
	ART	0.491	0.620	0.556 (−5.3%)
Default		0.543	0.631	0.587

Gene name identification. Using exact word matching for gene name identification severely deteriorated the performance on both of the training and test data sets. This supports our observation that gene names are often written in slightly different forms from their canonical ones (database entries). Thus, flexible name matching schemes such as the one tested here are needed in order to exhaustively locate gene name occurrences. A possible drawback of approximate word matching is that it may recognize irrelevant word sequences as gene names (i.e., false positives), leading to an inclusion of irrelevant text fragments into the representation of the target gene. However, the influence can be minimized by tuning the threshold (and parameters) for the word-overlap score defined in Equation (1). According to our experiments, a threshold of 0.3 (which was used for our experiments) constantly yielded the best performance.

Gene name dictionary. Not using the gene name dictionary also deteriorated classification both on the training and test data by 24.5% on average. It verifies that gene name expansion using the dictionary did help to identify text fragments relevant to the target gene, even though the dictionary was automatically compiled without manual curation.

Glossary and keyword fields. Contrary to our expectation, the use of glossary and keyword fields to search for gene synonyms was not found helpful for GO annotation. There was little or no difference between the F_1 scores produced with and without the use of the fields. Close examination revealed that these fields hardly provided information regarding gene synonyms and thus had little effect on the classification performance. To be exact, there were only 15 pairs of gene and synonyms found in these fields out of 882 articles in the training and test data sets.

MeSH terms. Similarly to the case of glossary and keyword fields, the F_1 scores show little or no difference between the settings where MeSH terms were used (Default) and unused (MeSH-unused). However, the difference becomes more apparent when looking at precision and recall. On the test data, Default and MeSH-unused yielded nearly equal precision (0.503 and 0.509, respectively), whereas using MeSH terms (Default) achieved a recall approximately 5% higher than MeSH-unused (0.847 and 0.808, respectively). It suggests that the inclusion of MeSH terms led to predict more potential classes for a given gene (which raised recall), but also produced some amount of false positives (which slightly decreased precision). The lower precision may be due to the fact that MeSH terms are not gene-specific.

Unit of extraction. Four different units were tested in extracting text fragments. In short, as going from *G* (only sentences containing target genes) to *ART* (entire articles) in Table 2, more text was extracted for document representation. As can be seen, there is a trend that F_1 slightly increases from *G* to *P*+*G*+*S* (sentences containing target genes plus immediately preceding and succeeding sentences) and then decreases when entire articles were used (i.e., *ART*) compared to Default. However, we should not overlook that, surprisingly, *ART* performed comparably to Default on the test data, suggesting that our framework to use only text fragments containing target genes is not necessarily very effective. Possible reasons are that (a) target genes were found everywhere in associated articles so that almost all paragraphs were extracted, making little difference whether entire articles or paragraphs were used; (b) not many articles were associated with multiple genes, and thus gene-specific document representation (such as ours) was not very important for the test data; and (c) multiple genes associated with single articles actually had almost the same classes. We investigated each possibility by comparing the training and test data but there was no noticeable difference found between them. To closely examine the effects of using paragraphs, we plotted recall-precision curves by varying the threshold for *k*NN (where the same thresholds were applied to all classes) as shown in Figure 3.
The top two curves were obtained on the test data and the bottom two were based on the training. Although it is less apparent compared to the case of training data, it can be seen that, overall, using paragraphs (shown as solid lines) marginally improved the performance also on the test data.

4.2.3 Contributions of different parts of articles

Our current framework, to some extent, takes into account the logical structure of input in a sense that it makes use of paragraph boundaries in extracting text fragments containing target genes. However, it does not consider or distinguish the structure of an article, e.g., sections. Such information may be useful for GO annotation because different parts of articles may have different im-

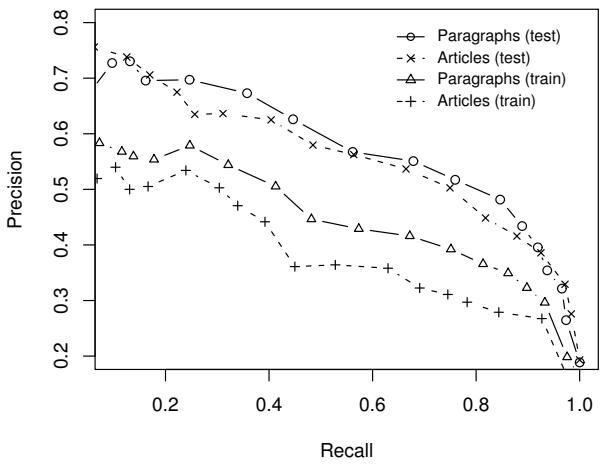

Figure 3: The relation between recall and precision where either selected paragraphs or entire articles were used for document representation.

portance with respect to GO annotation. For example, result and conclusion sections may be more relevant to GO annotation as they usually report findings from experiments. Therefore, we examined how useful the individual sections were for GO annotation by using only one section at a time from which gene-bearing paragraphs were extracted. Specifically, we focused on the following sections: abstract, introduction, procedures, methods, and results. Both discussion and conclusion sections were regarded as result sections since they are sometimes not clearly separated from results (e.g., "Results and Discussion" section). Incidentally, these sections were identified based on section names annotated by SGML tags.

Figure 4 shows a histogram for F_1 scores produced using single sections on the training data, where we include results from the use of only titles and only MeSH terms for comparison. The rightmost bar "All" used all the sections including MeSH, which corresponds to "Default" in Table 2.

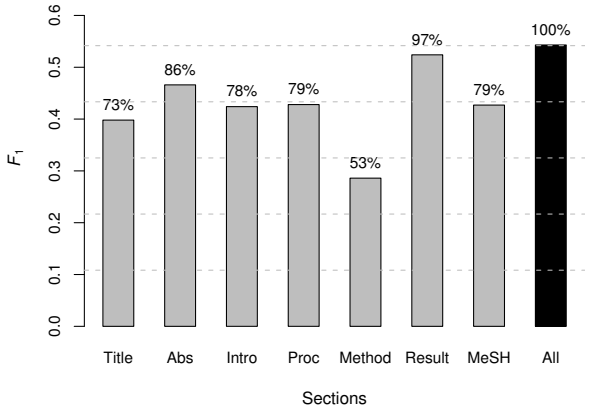

Figure 4: Results produced by individual sections. "All" used all the sections. Percentages above bars indicate the respective proportions to "All".

Surprisingly, the Result section alone yielded almost as good F_1 as All, followed by Abs (abstract), MeSH, Proc (procedures), and so on. On the other hand, the Method sections showed the least performance for GO annotation. Although not presented here, experiments on the test data also showed similar results.

4.3 The triage task

4.3.1 Overview

The framework we described above aimed at GO annotation. However, it can be also applied to another task from the TREC Genomics Track, i.e., the *triage task* (see Section 1). In brief, the triage task is to determine if an input article contains experimental evidence that warrants GO annotation, where no particular gene is specified. This task can be naturally regarded as a binary text categorization problem.

In terms of text categorization, a primary difference between GO annotation tackled in the previous sections and the triage task is that the former takes a pair of article and gene as input, whereas the latter takes only an article. As input is not gene-specific, the triage task could simply rely on an entire article for document representation without the necessity to locate the text fragments containing a particular gene. Yet, because the triage decision must be made in consideration of the genes mentioned in a given article, our framework to use only gene-bearing paragraphs may be more appropriate. Thus, we adapted our system to extract paragraphs that were likely to contain *any* gene names identified by the gene name recognizer YAGI [15]. Note that MeSH terms associated with a given article were also included as features as in GO annotation.

4.3.2 Methods

We used the same methods described in Section 2 for document representation and classification except the followings:

- For document representation, only paragraphs that were likely to contain any gene name (determined by YAGI) were used. Feature selection and term weighting were done in the same ways as GO annotation.

- For classification, the variant of *k*NN classifiers defined in Equation (6) was used but had only two classes, i.e., positive (POS) and negative (NEG). If an input article was classified as POS, it was outputted as positive irrespective of whether classified as NEG.

4.3.3 Data and evaluation measures

We used the TREC data sets provided for the triage task, which was composed of 5,837 full text articles for training (375 positives and 5,462 negatives) and 6,043 for test (420 positives and 5,623 negatives). As is the case with GO annotation, the training data were used for tuning parameters and were used as pre-labeled instances for *k*NN to classify the test data. Specifically, the number of neighbors k and the value of threshold t_{POS} were set to 160 and 93.4, respectively, which produced the best result in normalized utility measure (explained next) on the training data.

For the evaluation measure, normalized utility measure U_{norm} defined below was used according to the TREC evaluation.

$$U_{norm} = U_{raw}/U_{max} \tag{8}$$

where

$$\begin{aligned} U_{raw} &= 20 \times TP - FP \\ U_{max} &= 20 \times (TP + FP) \end{aligned} \tag{9}$$

TP and *FP* denote the number of articles correctly identified as positive (true positive) and the number of articles falsely identified as positive (false positive), respectively.

4.3.4 Results and discussions

Table 3 compares our results with the representative results from the TREC official evaluation, where precision and recall results are also presented.

Table 3: The TREC official results and our results for the triage task (on the test data set).

		Prec	Recall	U_{norm}
TREC	Best	0.157	0.888	0.651
	Worst	0.200	0.014	0.011
	Mean	0.138	0.519	0.330
Ours	TFIDF	0.112	0.752	0.455
	$TFCHI_1$	0.160	0.883	0.651
	$TFCHI_2$	0.137	0.826	0.567

Our framework with the term weighting scheme $TFCHI_1$ compared favorably with the best performing system reported at TREC, while TFIDF did not perform as well. This is mainly because the TFIDF scheme could not assign an appropriate (high) weight to the MeSH term "`Mice`" since it appeared in many documents (leading to a low IDF value). It was reported that a simple rule which classifies articles annotated with the MeSH term `Mice` as positive and those without it as negative could achieve nearly as good performance as the best reported result [1].

Unlike TFIDF, TFCHI considered word distributions across different classes and was able to assign higher weights even to the terms that appeared in many documents but almost only within a class, such as `Mice` in this particular data sets. To contrast the difference between IDF and χ^2 values, we plotted a scatter diagram for corresponding IDF and χ^2 as shown in Figure 5, where MeSH terms are indicated by capitalization.

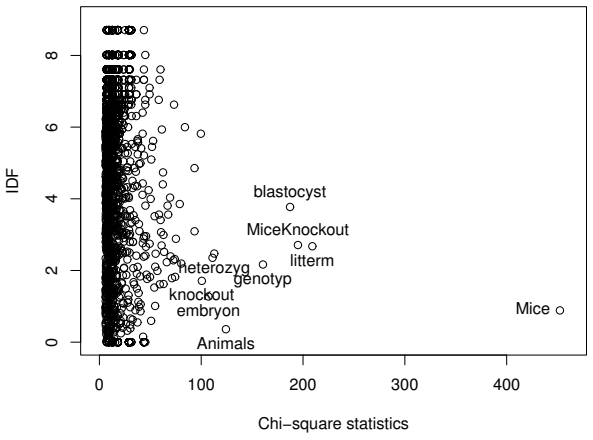

Figure 5: A scatter plot for χ^2 and IDF.

As can be seen, high-χ^2 words, such as `Mice`, `Animals`, `embryon` (the stem for embryonic), and `knockout`, were not necessarily assigned high IDF values. Interestingly, the correlation coefficient for IDF and χ^2 turned out to be -0.13, which is usually strongly and positively correlated as empirically known in the text categorization literature [19]. The result suggests that TFIDF, which is often used for text categorization, is not necessarily optimum depending on the characteristics of the target data. This supports the idea of the supervised term weighting schemes [2] that class-based term weights (e.g., chi-square statistics) is more appropriate for classification.

It may be also possible, however, that our framework with $TFCHI_1$ performed well solely because of the notably high value of χ^2 associated with the MeSH term `Mice` (remember that simple heuristics using `Mice` could perform very well). To investigate, we applied the $TFCHI_1$ scheme to hypothetical test data where the MeSH term `Mice` was completely removed. The resulting normalized utility score was 0.548, which outperforms the TFIDF scheme in Table 3 and is still comparable to the second best system [4] (which produced a U_{norm} of 0.549) in the TREC evaluation.

5. RELATED WORK

This section discusses representative work by other researchers for the GO annotation and triage tasks.

For GO annotation, Settles et al. [16] developed a two-tier classification framework using Naïve Bayes (NB) classifiers and Maximum Entropy (ME) models with several external resources and specialized features. They exploited the structure of articles and distinguished six section types (such as introduction and discussion) as a unit of classification. They created an NB classifier for each section and the output probabilities of the NB classifiers were then combined using ME models which differently weighted each of the section types and classes. The features used for the NB classifiers included not only words from body text but also syntactic patterns and what they call informative terms. The syntactic patterns are frequent patterns for subjects and direct objects (e.g., "translation of *X*") automatically collected from training data using a shallow parser. The informative words were word *n*-grams ($1 \le n \le 3$) having high chi-square statistics. To supplement the relatively small size of the training data provided by TREC, they used external resources including the BioCreAtIvE data set and MEDLINE abstracts with which specific genes and GO codes were associated in existing databases other than MGI. The reported F_1 score was 0.514, which is 14% lower than our best score reported here. The difference is presumably due to the fact that their system did not employ gene name expansion and approximate word matching which we found highly important for GO annotation.

For the triage task, Dayanik et al. [1] applied Bayesian logistic regression (BLR) models, which estimate a probability that an input belongs to a specific class. For document representation, they used MeSH terms from the MEDLINE database in addition to input articles. Their best result was achieved by applying the following configuration. They used *only* title, abstract, and MeSH terms for features and applied the conventional TFIDF term weighting scheme, and proposed a two-stage classifier which assigned negative to all articles *not* indexed with the MeSH term `Mice` and classified those indexed with `Mice` by using BLR. The reported normalized utility score is 0.641. In spite of using TFIDF, which yielded suboptimal results in our experiments, their method outperformed other TREC participants.

6. SUMMARY AND FUTURE RESEARCH

This paper presented our work on automating GO domain code annotation. We approached this task by treating it as a text categorization problem and adopted a variant of *k*NN classifiers. To apply *k*NN, we first represented each input, $\langle \text{article}, \text{gene} \rangle$ pair, by a term vector, where terms were collected from text fragments (paragraphs) containing the target gene. To exhaustively locate the gene name occurrences, we took advantage of existing databases to

automatically compile a gene name synonym dictionary and pre-processed both gene names and text to tolerate minor differences between them. In addition, we utilized approximate word matching to identify gene occurrences to deal with other irregular forms of the gene names. The collected words were then fed to feature selection using chi-square statistics, which were re-used for term weights adopting supervised term weighting schemes. We evaluated the proposed framework on the TREC Genomics Track data sets and showed that, overall, our method performed the best compared with the TREC official evaluation. Further analyses revealed that the flexible gene name matching used in conjunction with the gene name dictionary was notably effective. Another finding is that the result sections of articles contributed the most for GO annotation. It was also demonstrated that our framework was successfully applied to a related but different problem, the triage task, producing a normalized utility score of 0.651 which is comparable to the best reported performance at the recently held TREC. In addition, the TFIDF scheme was found suboptimal for this particular task and data sets.

For future research, we are planning to explore a better use of structure of articles (e.g., sections) and local context around the target genes. Such information may be incorporated into the current framework by way of term weights. Another direction is to extend our work to more advanced, realistic settings. For example, in the real-world GO annotation, genes are not given in advance. Taking only articles as input without specific genes would be an interesting challenge.

7. ACKNOWLEDGMENT

This project is partially supported by the NSF grant ENABLE #0333623.

8. REFERENCES

[1] A. Dayanik, D. Fradkin, A. Genkin, P. Kantor, D. D. Lewis, D. Madigan, and V. Menkov. DIMACS at the TREC 2004 genomics track. In *Proceedings of the 13th Text REtrieval Conference (TREC 2004)*, 2004.

[2] Franca Debole and Fabrizio Sebastiani. Supervised term weighting for automated text categorization. In *Proceedings of SAC-03, 18th ACM Symposium on Applied Computing*, pages 784–788, 2003.

[3] Sergei Egorov, Anton Yuryev, and Nikolai Daraselia. A simple and practical dictionary-based approach for identification of proteins in MEDLINE abstracts. *Journal of the American Medical Informatics Association*, 11(3):174–178, 2004.

[4] Sumio Fujita. Revisiting again document length hypotheses TREC-2004 genomics track experiments at Patolis. In *Proceedings of the 13th Text REtrieval Conference (TREC 2004)*, 2004.

[5] Daniel Hanisch, Juliane Fluck, Heinz-Theodor Mevissen, and Ralf Zimmer. Playing biology's name game: Identifying protein names in scientific text. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, volume 8, pages 403–414, 2003.

[6] William Hersh. Text retrieval conference (TREC) genomics pre-track workshop. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, page 428, 2002.

[7] William Hersh. Report on TREC 2003 genomics track first-year results and future plans. *SIGIR Forum*, 38(1):69–72, 2004.

[8] W.R. Hersh, R.T. Bhuptiraju, L. Ross, A.M. Cohen, and D.F. Kraemer. TREC 2004 genomics track overview. In *Proceedings of the 13th Text REtrieval Conference (TREC 2004)*, 2004.

[9] Lynette Hirschman, Jong C. Park, Jun-ichi Tsujii, Limsoon Wong, and Cathy H. Wu. Accomplishments and challenges in literature data mining for biology. *Bioinformatics*, 18(12):1553–1561, 2002.

[10] Julie Beth Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

[11] Claire O'Donovan, Maria Jesus Martin, Alexandre Gattiker, Elisabeth Gasteiger, Amos Bairoch, and Rolf Apweiler. High-quality protein knowledge resource: SWISS-PROT and TrEMBL. *Brief Bioinform*, 3(3):275–284, 2002.

[12] Kim D. Pruitt and Donna R. Maglott. RefSeq and LocusLink: NCBI gene-centered resources. *Nucleic Acids Research*, 29(1):137–140, 2001.

[13] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1983.

[14] Ariel S. Schwartz and Marti A. Hearst. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, volume 8, pages 451–462, 2003.

[15] Burr Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA)*, 2004.

[16] Burr Settles and Mark Craven. Exploiting zone information, syntactic rules, and informative terms in gene ontology annotation of biomedical documents. In *Proceedings of the 13th Text REtrieval Conference (TREC 2004)*, 2004.

[17] Hagit Shatkay and Ronen Feldman. Mining the biomedical literature in the genomic era: An overview. *Journal of Computational Biology*, 10(6):821–856, 2003.

[18] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, 1999.

[19] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420, 1997.