Feature generation, feature selection, classifiers, and conceptual drift for biomedical document triage

A. M. Cohen, R.T. Bhupatiraju, W.R. Hersh
Department of Medical Informatics and Clinical Epidemiology, Oregon Health & Science University, Portland, OR, USA

ABSTRACT

We approached the problem of classifying papers for the TREC 2004 Genomics Track triage task as a four step process: feature generation, feature selection, classifier training, and finally, classification. Section specific binary features that discriminated significantly between positive and negative training samples were chosen using the Chi-square statistic. Three classifiers were trained on this feature set: a simple Naive Bayes classifier, the SVMLight support vector machine implementation, and a voting perceptron extended to support variable learning rates. Comparing the classifiers on the training data we found that neither Naive Bayes nor SVMLight was able to adequately account for the factor of 20 in the utility function. The voting perceptron classifier performed much better at this. The performance on the test collection was lower for all classifiers, although consistent with the relative values of the training cross-validation. Feature subsetting showed no significant differences in precision or recall, implying that there was some redundancy among the features. We also examined how well the feature set derived from the 2002 training collection represented the papers in the 2003 test collection, and found a low level of similarity between feature sets derived from the two collections. This supports the hypothesis that important classification terms change quickly over time.

1 INTRODUCTION

The 2004 Text Retrieval Conference (TREC) Genomics track was divided into two main tasks: categorization, and ad-hoc retrieval. The categorization task was composed of a document triage subtask and an annotation subtask to detect the presence of evidence in the document for each of the three main Gene Ontology (GO) code hierarchies. Our work focused on the document triage subtask. We also participated in the ad-hoc retrieval task

2 BACKGROUND

Document classification is a common problem in biomedicine. Training a support vector machine (SVM) on vectors created from stemmed and/or stopped document word counts has proven to be a basic and typically successful approach (Yeh *et al.*, 2003). However we believed that the triage problem posed here had several distinctive features that would require modification to the basic approach.

First, the number of true positives in both the training and test collection was known to be small, between 6-7%. Second, the utility function chosen as the metric of record was heavily weighted to reward recall and not precision. This was based on an analysis of the current working procedures of the annotators at the Mouse Genome Institute (MGI), and an approximation of how they currently value false negative and false positive classification. The official utility function weights a false negative as twenty times more serious than a false positive. By this measure the current work practice of MGI, which is read all documents in test collection, has a utility of 0.25.

Additionally, the training and test collections were not randomly drawn from the same sample but instead were collected from documents published in two sequential years. While this is a more realistic simulation of a system as it would be put into use at MGI, it raises the issue of how well the feature set derived from one year of literature represents the literature of subsequent years.

Because of these issues our approach included a rich set of feature types, statistically based feature selection, several classifiers, and an analysis of how well the feature set derived from the year 2002 corpus represented the documents in the 2003 corpus.

3 SYSTEM AND METHODS

We approached the triage problem in four stages: Feature generation, feature selection, classifier selection and training, and finally, test document classification. The first three steps were performed using only the training set. The final step was performed on the test collection to generate the submitted results.

During system development we used ten-fold cross-validation on the training set to compare approaches and set system parameters. This entailed performing the first two steps on the entire training collection. Then 90% of the training data was used to train the classifiers which were then applied to the remaining 10% of the training data. This was repeated nine more times, so that all of the training data was classified once. The results were then aggregated to compute cross validation metrics on the training corpus. Figure 1 presents this process diagrammatically.

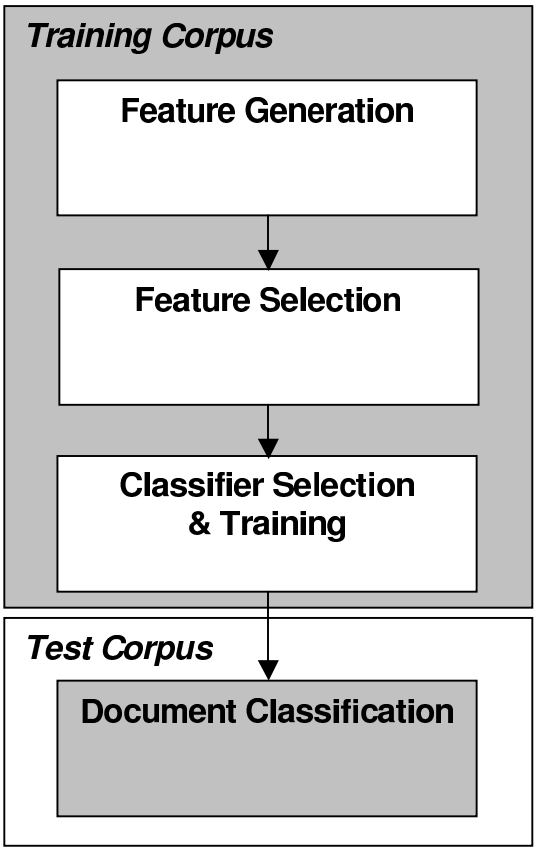

Fig. 1. Step-wise approach to text classification

3.1 Feature generation

The full text corpus with SGML mark-up provided opportunity to investigate the use of many types of features. While many text classification approaches treat the text as a "bag-of-words", we chose to use the information contained in the SGML mark-up to generate section type specific features. Because we combined features that could occur multiple times in a single document with features that could occur only once, after some initial testing we choose to treat each feature as binary, that is, each feature was either present in a document or it was absent.

One type of feature that we generated consisted of pairs of section names and stemmed words, using the Porter stemming algorithm. After applying a stop list of the 300 most common English words, individual parts of the document processed were processed to include sections for the abstract, body paragraphs, captions, and section titles. We also created similar combination section, stemmed word features using the stopped and stemmed section title in combination with the stopped and stemmed words within the named section.

In addition we downloaded the corresponding MEDLINE records from PubMed. The corresponding MeSH headings were extracted for each article. We included MeSH-based feature types based on the full MeSH headings, the MeSH main headings, and the MeSH subheadings.

Finally we included feature types based on information in the references section of each document. The main author of each reference was taken as a feature type. We also included long form of references as a feature type, by including the primary author, the journal name, volume, year, and page number.

Running the feature generation process on the full set of 5837 training documents produced over 100,000 potentially useful features along with counts of the number of documents containing each feature.

3.2 Feature selection

We chose to use the Chi-square selection method to select the features that best discriminated between positive and negative documents in the training corpus. The 2x2 Chi-square table is constructed as shown in Table 1, using the document counts obtained in the previous step. During system tuning it was found that an alpha value of 0.025 produced the best results. Using this value as a cut-off, 1885 features were selected as the most significant. The number and type of each feature found significant and used in the following steps are shown in Table 2.

3.3 Classifier selection and training

We applied three different classifiers to the problem: Naive Bayes, SVM, and Voting Perceptron. While it is commonly thought that the best classifiers are based on the SVM approach of Vapnik (Vapnik, 2000), the distinctive aspects of the current classification problem discussed above motivated us to apply three different classifiers. By using the same feature set with each of the classifiers, this allowed us to compare the effectiveness of the classifier algorithms for the particular requirements of the triage task.

Table 1. 2x2 arrangement for testing feature significance

		Feature is the one under test?	
		Yes	**No**
Training document is triage positive?	**Yes**	Number of times feature seen in positive documents	Number of times other features seen in positive documents
	No	Number of times feature seen in negative documents	Number of times other features seen in negative documents

Table 2. Number and type of features used

Feature type	Number significant
Abstract stemmed words	127
Body paragraph stemmed words	778
Caption stemmed words	291
MeSH full headings	15
MeSH main headings	52
MeSH subheadings	5
Author of referenced work	35
Reference	4
Section title stemmed words	69
Section title with stemmed section words	509
Total number of features significant features	**1885**

Neither Naive Bayes, nor the implementation of SVM we used, SVMLight (Joachims, 2004), provided adequate means of adjusting for the low frequency of positives and the high utility of true positives relative to false positives. For Naive Bayes, we used our own implementation. Naive Bayes provides a classification probability threshold that can be used to trade off between precision and recall. However, this is an indirect means of compensation, and in practice for this classification task, we found adjusting the probability threshold did not have a significant effect.

We fully expected SVMLight to perform better than Naive Bayes, since it included a cost factor parameter that can be adjusted to allow unequal penalties for false positives and false negatives. However, we found that the amount of influence this parameter has is small, and was inadequate to compensate for the factor of 20 difference between the cost of false positives and negatives. Since neither Naive Bayes nor one of the most popular implementations of SVM addressed our requirements, something else was needed.

A review of the classification literature shows significant work in modifying the classic Perceptron algorithm of Rosenblatt (Rosenblatt, 1958) to achieve performance at or near that of SVM for many problems. One algorithm in particular, the Voting Perceptron algorithm (Freund and Schapire, 1999), has very good performance, is quite fast, and is easy to implement. While the algorithm as published does not include a means of compensating for asymmetric false positive and negative penalties, we have created a modification to the algorithm that does.

A perceptron is essentially an equation for a linear combination of the values of the feature set. There is one term in the perceptron for each feature in the feature set, plus an optional bias term. A document is classified by taking the dot product of a document's feature vector with the perceptron, and adding in the bias term. If the result is greater than zero, then the document is classified as positive, if less than or equal to zero, then the document is classified as negative.

Rosenblatt's original algorithm trained the perceptron by applying it to each sample in the training data. If the sample was incorrectly classified, the perceptron was modified by adding or subtracting the a sample back into the perceptron, adding when the sample was a true positive, and subtracting when the sample was a true negative. Over a large number of training samples the perceptron converges on the solution that best approximates the separation between positive and negative documents in the training set.

Freund and Schapire improved the perceptron's performance by modifying the algorithm to produce a series of perceptrons, each which makes a prediction on the class of each document and gets a number of "votes" depending upon how many documents that perceptron classified correctly in the training set. The class with the most votes is the predicted class assigned to the document.

Our extension to this algorithm is based upon adjusting the learning rate of the perceptron differently for false negatives and false positives. While in the typical implementation, incorrectly classified samples are directly added or subtracted back into the perceptron, we first multiply the sample by a factor known as the *learning rate*. Furthermore, we use different learning rates for false positives and false negatives. Given the definition of the utility function, we expected that the optimal learning rate for false negatives to be about 20 times that of false positives. This is indeed what we found during training. We used a learning rate of 20.0 for false negatives, and 1.0 for false positives.

Each of the three classifiers was applied to the training corpus. Ten-fold cross-validation was used to optimize any free parameters. The Naive Bayes classifier had one free parameter, the probability classification threshold. This was left at the default value of 0.50. The SVM-Light classifier settings chosen used the linear kernel and a cost factor of 20.0. The Voting Perceptron classifier was used with a linear kernel, and the learning rates were given above. This created a trained classifier model for each of the three methods.

3.4 Classification of test documents

Finally, the models created by the Naive Bayes, SVM, and Voting Perceptron classifiers were applied to the test corpus. This is done in two steps. First the documents in the test corpus were analyzed for the presence or absence of the significant features found during the feature selection step. This created a feature vector for each test document. Then the documents were classified by applying each of the three trained classifiers.

Table 3. Performance of classification system on test and training corpi

Corpus	Classifier	Precision	Recall	Fscore	Utility
Training corpus	Naive Bayes	0.1556	0.7650	0.2587	0.5577
	SVMLight	0.3140	0.5550	0.4010	0.4940
	Voting Perceptron	0.1857	0.8453	0.3045	0.6600
Test corpus	Naive Bayes	0.1290	0.6548	0.2155	0.4337
	SVMLight	0.2309	0.3524	0.2790	0.2937
	Voting Perceptron	0.1714	0.6571	0.2719	0.4983

3.5 Evaluation of conceptual drift

One important issue in applying text classification systems to documents of interest to curators and annotators is how well the available training data represents the documents to be classified. When classifying a biomedical text, the available training documents must have been written before the text to be classified. However, by its very nature the field of science changes over time, as does the language used to describe it. How rapidly the written literature of science changes has a direct influence on the development of biomedical text classification systems in terms of how features are generated and chosen, how often the systems need to be retrained, how large the training increment should be, and may effect the maximum performance that can be expected out of these systems.

We wanted to begin to understand this important issue of *conceptual drift* in the biomedical literature. In order to measure how well the features chosen from the training collection represented the information important in classifying the document in the test collection, we performed additional steps of feature generation and selection on the test collection. The exact same system and parameters were applied to the test collection as the training collection. Then we measured how well the training collection feature set represented the test collection feature set by computing similarity metrics between the two sets (Dunham, 2003).

4 RESULTS

4.1 Classification performance

The results of applying our classification systems to both the training and test collections are presented in Table 3. As previously stated, ten-fold cross validation was used to evaluate performance on the training data. The results on the test data were created blindly, running the algorithms on the test corpus and sending the results in for evaluation by the TREC 2004 Genomics Track staff.

As can be seen in the table, the Voting Perceptron algorithm had the best utility of 0.6600, Naive Bayes next, and SVMLight worst on both the training and test corpi. The highest recall obtained was 0.8453 by the Voting Perceptron algorithm on the training collection. The highest precision of 0.3140 was obtained by SVMLight on the training collection, and the highest overall F-score was obtained by SVMLight also on the training collection.

The same relative performance was obtained for all three algorithms for precision, recall, F-score, and utility on the test collection, however the actual numbers are much lower. Voting Perceptron utility fell to 0.4983, and recall fell to 0.6571. SVMLight's precision and F-score fell to 0.2309 and 0.2790 respectively. Also the F-score of the Voting Perceptron algorithm on the test data was 0.2719, almost equal to that of SVMLight.

Figure 2 presents the results of applying the Voting Perceptron classifier to the training data using ten-fold cross validation and leaving out feature types. Each vertical row shows the recall and precision obtained when leaving out one type of feature. For comparison, the leftmost column shows the performance of the full feature set. No significant differences in recall or precision were found by leaving out single feature types. This may indicate that there is redundancy in the feature set. In fact there is some textual overlap between the body paragraph stemmed words feature type and the section title with stemmed section words feature type, and also between the author of referenced work and reference feature types.

4.2 Conceptual drift

Because the metrics on the test collection are much lower than the cross validation on the training set, it useful to understand how well the feature set extracted from the training collection represents the test collection. We performed the same process of feature generation and selection on the test collection. The process generated a set of 1899 significant features, a quantity very close to the 1885 features extracted from the training collection.

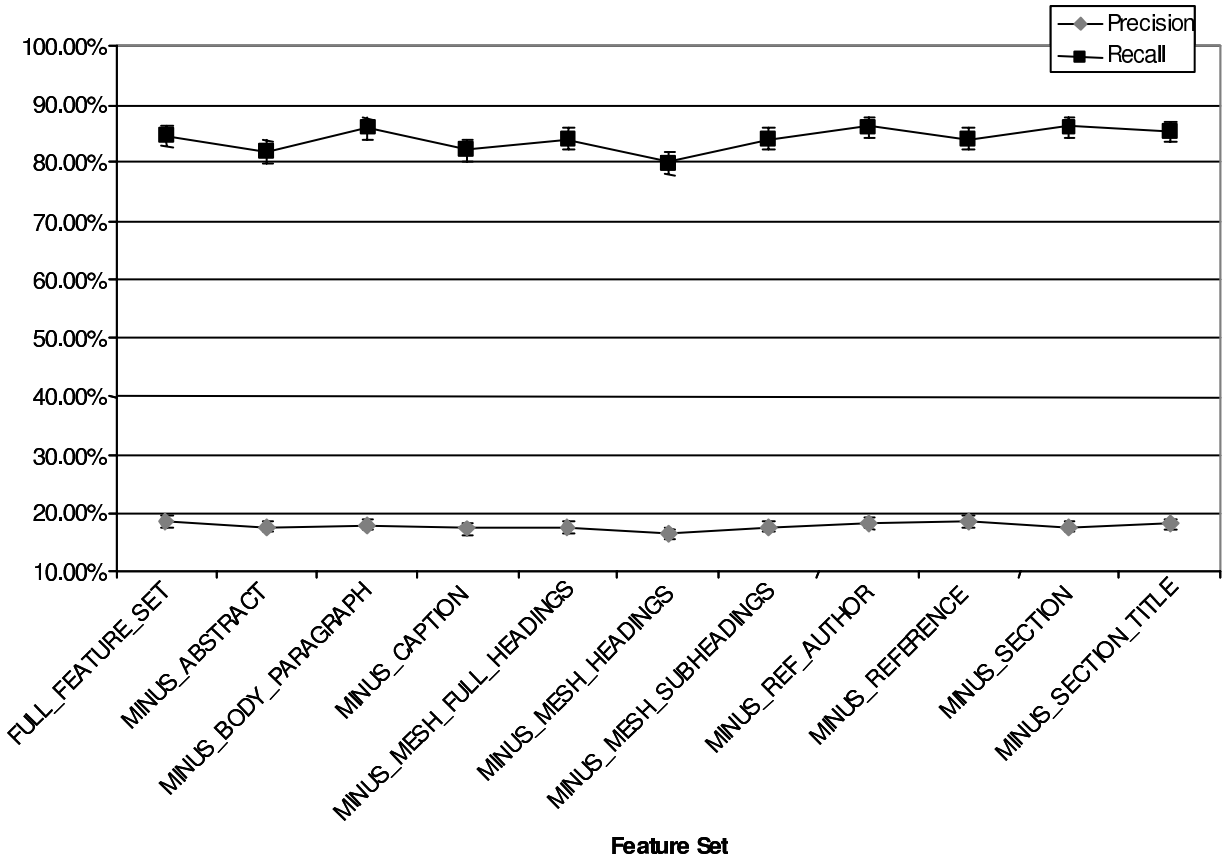

Fig. 2. Precision and recall when leaving out feature types

We computed similarity measures between these two sets of features. The Dice similarity coefficient was 0.2489, the Jaccard similarity was 0.1422, cosine similarity was 0.2489, and the overlap measure was 0.2499. All similarity measures showed a low level of similarity between the two sets.

This conceptual drift is not simply a reflection of a wholesale change in vocabulary. We performed equivalent similarity measures on the individual word frequencies in the training and test collection, filtered out common English words as before, and sorted the words most frequent to least frequent for both sets. Computing similarity measures between the top 100, 1000, and 10,000 words in both sets showed consistently high similarity measures, with the maximum being the Dice similarity coefficient of 0.9618 at 100 words, and the minimum being a Jaccard similarity of 0.9232 at 10,000 words.

5 DISCUSSION

These results show that document classification can be a useful technique to aid the MGI curators in screening documents for annotation. The utility of the Voting Perceptron system on the test collection is about twice that of the estimate of the current work practice of MGI, and is ~14% better than the next best classifier. Clearly it is important for biomedical document classifiers to be able to flexibly incorporate utility measures specific to the task, as we have done here with the Voting Perceptron classifier.

Nevertheless, the performance measures on the test collection are significantly lower than those on the training set. One possible explanation for this can be found in the low similarity between the two sets of significant features extracted for the two text corpi. The maximum similarity metric, the overlap measure, only shows about 25% overlap between the two sets. Therefore, many of the features found significant during training were in fact not significant for triaging the test collection.

There may be many factors influencing why this is so. As the vocabulary similarity measures show, it is not

simply a wholesale change in the language used in journal articles. The cause is something more subtle, and more specific to the terms and concepts that are important in the classification of these documents for annotation triage.

One possible explanation is that the important words and concepts that signify inclusion in the positive triage set changed between the years during which the documents in the test and training sets were written, 2002 and 2003 respectively. This may be due to authors using new concepts or different language. It may also signify that the criteria used by the annotators when triaging a document has changed.

Clearly this issue needs more study if we are to apply text classification in a manner that best addresses the needs of annotators. Document triage systems may need to be re- trained more frequently, or even continuously trained. It may also be important to develop methods of extracting sets of features that have greater longevity than the Chi-square method used here.

6 CONCLUSIONS

Automated document triage as presented here can be a useful aid to the MGI triage process. The current state of the art provides a notable increase in utility above the current work process. However, more work needs to be done to verify that the utility metric used here actually represents value as perceived by the MGI curators. Furthermore, the best means of deriving and updating the classification feature set over time is an open question and needs further study.

7 AD-HOC INFORMATION RETRIEVAL TASK

OHSU also took part in the ad hoc retrieval task of the Genomics Track. For the task, we decided to see how known simple but effective indexing and retrieval strategies would fare with the test collection. As such, we used the Lucene system, which is part of the Jakarta open source distribution of Web tools. Lucene implements a variant of TF*IDF term weighting that includes additional parameters for query term boosting and document length normalization (Apache Software Foundation, 2004). We did not use boosting and manual inspection showed length normalization to be detrimental. Therefore our runs were based on TF*IDF term weighting for document ranking.

We submitted two official runs, one that used just the information needs statement of the query (OHSUNeeds) and the other, which used all of the text, including the title, information need, and context (OHAUAll). Both of these runs performed above the median in mean average precision (MAP), with the OHSUNeeds run scoring slightly better. Determining why these simple approaches worked better than many others requires further analysis, but could be due to more elaborate methods having detrimental effects.

ACKNOWLEDGEMENTS

This work was supported in part by Grant number 2 T15 LM07088-11 from the National Library of Medicine, and Grant ITR-0325160 from the National Science Foundation.

REFERENCES

Apache Software Foundation (2004) Class Similarity, Lucene 1.4.2 API. http://jakarta.apache.org/lucene/docs/api/org/apache/lucene/search/Similarity.html

Dunham, M. H. (2003) Data mining introductory and advanced topics. Prentice Hall/Pearson Education, Upper Saddle River, N.J.

Freund, Y. and Schapire, R. E. (1999) Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, **37**, 277-296.

Joachims, T. (2004) SVM-Light Support Vector Machine. http://svmlight.joachims.org/

Rosenblatt, F. (1958) The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 386–407.

Vapnik, V. N. (2000) The nature of statistical learning theory. Springer, New York.

Yeh, A. S., Hirschman, L. and Morgan, A. A. (2003) Evaluation of text data mining for database curation: lessons learned from the KDD Challenge Cup. *Bioinformatics*, **19 Suppl 1**, i331-9.