Journal of Data Science **5**(2007), 259-267

Automated Linking PUBMED Documents with GO Terms Using SVM

Su-Shing Chen and Hyunki Kim
University of Florida, Gainesville

Abstract: We have developed an automated linking scheme for PUBMED citations with GO terms using SVM (Support Vector Machine), a classification algorithm. The PUBMED database has been essential to life science researchers with over 12 million citations. More recently GO (Gene Ontology) has provided a graph structure for biological process, cellular component, and molecular function of genomic data. By text mining the textual content of PUBMED and associating them with GO terms, we have built up an ontological map for these databases so that users can search PUBMED via GO terms and conversely GO entries via PUBMED classification. Consequently, some interesting and unexpected knowledge may be captured from them for further data analysis and biological experimentation. This paper reports our results on SVM implementation and the need to parallelize for the training phase.

Key words: Classification, gene ontology, support vector machines.

1. Introduction

With the exponential growth of biomedical data, life science researchers have met a new challenge - how to exploit systematically the relationships between genes, sequences and the biomedical literature (Yandell and Majoros, 2002). Usually most of known genes are found in the biomedical literature and PUBMED is a worthy database for this kind of information. PUBMED, developed by the U.S. National Library of Medicine (NLM), is a database of indexed bibliographic citations and abstracts (National Library of Medicine). It contains over 4,600 biomedical journals. PUBMED citations and abstracts are searchable via PUBMED[1] or the NLM Gateway[2]. The biomedical literature has much to say about gene sequence, but it also seems that sequence can tell us much about the biomedical literature. Currently, highly trained biologists read the literature and

[1]http://www.ncbi.nlm.nih.gov/entrez/query.fcgi
[2]http://gateway.nlm.nih.gov/gw/Cmd

manually select appropriate Gene Ontology (GO) terms to annotate the literature with GO terms. Gene Ontology database has more recently been created to provide an ontological graph structure for biological process, cellular component, and molecular function of genomic data (Smith *et al.*, 2003).

McCray *et al.* (2002) show that the GO is suitable as a resource for natural language processing (NLP) applications because a large percentage (79%) of the GO terms have passed the NLP parser. They also show that 35% of the GO terms were found in a corpus collected from the MEDLINE database[3] and 27% of the GO terms were found in the current edition of the Unified Medical Language System (UMLS). A recent research work of Raychaudri *et al.* employs a "maximum entropy" technique to categorize 21 GO terms using training and test documents extracted from PUBMED using handcrafted keyword queries. Their study reports that their models trained on PUBMED documents published prior to 2001 achieved an accuracy of 72.8% when tested on documents published in 2001 (Raychaudhuri *et al.*, 2002). Another work of T. C. Smith *et al.* completed in April 2003 shows that about 110,000 PUBMED abstracts can be linked to the Gene Ontology (Smith and Cleary, 2003). In order to compare with (Raychaudhuri et al., 2002), it ran on the same 21 categories achieved an accuracy of 70.5% (at the precision-recall breakeven point).

Although these research works demonstrate that NLP is applicable to GO and PUBMED database can be linked to GO terms, there are inherently challenging issues to fully exploit both PUBMED and GO databases. One of them is that there are too many class categories (i.e. GO terms) in the GO because the GO is a large, complex graph in itself. For example, in the GO database released as of February 2005, there were a total of 17,593 terms (Gene Ontology Consortium[4]). Furthermore, GO grows in coverage and evolves in a monthly cycle. Finally, PUBMED contains over 12 million article citations. Beginning in 2002, it began to add over 2,000 new references on a daily basis (National Library of Medicine).

In order to organize the PUBMED contents in a systematic and useful way, we believe that text classification and text clustering should be exploited extensively. Perhaps, due to the large scale of PUBMED, it is also important to look for parallel and scaling-up algorithms. Text classification is a "boiling down" of the specific content of a document into a set of one or more pre-defined labels (Hearst, 1999). Text clustering can group similar documents into a set of clusters based on shared features among subsets of the documents (Chakrabarti, 2000; Chen *et al.*, 1996; Kohonen, 1998). In this paper, we have implemented a text classification system using SVM that can automatically link PUBMED citations

[3]National Library of Medicine: MEDLINE Fact Sheet.
http://www.nlm.nih.gov/pubs/factsheets/medline.html

[4]See Gene Ontology Consortium: Gene Ontology. http://www.geneontology.org/.

with GO terms. The performance measure for three data sets of small, medium and large sizes is excellent except training time. Then we examine the scalability of the SVM algorithm for training time. From the performance results of the three dataset sizes, we conclude that SVM must be scaled up using grid computers for its most computation-intensive task: training.

2. Implementation

First we consider basic terminology of text classification. Given a fixed set of classes $C = \{c_1, \ldots, c_n\}$, a training instance is a pair (d_i, c_i), where d_i is a document represented as a vector of m attribute values $\overrightarrow{d}_i =< w_1, \ldots, w_m >$, depicting m easurements made on the vocabulary $V = \{W_1, \ldots, W_m\}$, and $c_{\in}C$ is a class label associated with d_i for a given instance. A training set S is a set of labeled instances $S = \{(v_1, c_1), \ldots, (v_m, c_m)\}$. The goal in document classification is to infer a classification rule from the training set S so that it classifies new examples with high accuracy (Joachims, 2001).

The naive Bayes (NB) classifier is a probabilistic classification method (Lewis, 1998). NB is based on the Bayes' theorem and the naive Bayes independence assumption. Bayes' theorem says that to achieve the highest classification accuracy, a document d should be assigned to the class c_i for which $P(c_i|d)$ is highest. The naive Bayes independence assumption states that the probability of a word w_i is independent from any other word w_j given that the cass is known. Although this assumption is clearly false, it allows the easy estimation of the conditional probability $P(W_j|c_i)$. In the learning phase, NB estimates the class prior probabilities $P(c_i)$ and the conditional probability of each attribute w_i given the class label $c_i P(W_j|c_i)$. The estimated $P(c_i)$ is

$$P(c_i) = \frac{|c_i|}{|S|}$$

where $|c_i|$ denotes the number of training documents in class c_i and $|S|$ is the total number of training documents.

Given a new document $d =< w_1, \ldots, w_m >$, NB predicts the class as the one with the highest probability of c_i:

$$P(W_j|c_i) = \frac{1 + TF(W_j, c_i)}{|V| + \sum_{W' \in V} TF(W', c_i)}$$

where $|V|$ is the total number of attributes in V and $TF(W, c_i)$ is the overall number of times word w occurs within the documents in class c_i.

At training time, NB requires linear time both to the number of training documents and to the number of features and thus its computational requirements

Table 1: Description of datasets

Dataset	GO Term	GO ID	#Train	#Test
Large dataset	homeostasis	GO:0001116	14919	6396
	tissue regeneration	GO:0012976	17542	7519
	intercellular junction	GO:0003722	17893	7670
	cytochrome	GO:0003314	16226	6955
	collagen	GO:0003403	14579	6250
	synapsis	GO:0005240	15113	6479
	memory	GO:0005781	15085	6466
	tumor antigen	GO:0005995	16871	7232
	drug resistance	GO:0007018	15620	6696
	cell differentiation	GO:0000885	16354	7011
Medium dataset	locomotion	GO:0012740	7303	3132
	wound healing	GO:0001321	6995	2999
	endocytosis	GO:0004992	8043	3449
	nucleotide-excision repair	GO:0000635	8212	3521
	DNA replication	GO:0000660	8534	3659
	sex chromosome	GO:0000748	7106	3047
	synaptic transmission	GO:0000894	7684	3295
	ciliary or flagellar motility	GO:0000944	7229	3100
	menstrual cycle	GO:0000949	7465	3210
	peptide cross-linking via an oxazole or thiazole	GO:0009882	7627	3261
Small dataset	glial fibrillary acidic protein	GO:0013861	913	393
	proliferating cell nuclear antigen	GO:0001690	917	395
	nuclear membrane	GO:0003460	914	393
	intermediate filament	GO:0003464	854	368
	endosome	GO:0003596	942	405
	fermentation	GO:0003973	844	363
	phosphorylation	GO:0003982	847	365
	spectrin	GO:0005892	834	359
	oogenesis	GO:0007709	910	391
	acrosome	GO:0001137	824	354

are minimal. At classification time, a new example can be also classified in linear time both to the number of features and to the number of classes. NB is particularly well suited when the dimensionality of the inputs is high and can often outperform more sophisticated classification methods due to its simplicity

and effectiveness (Liu *et al.*, 1998).

Support Vector Machine (SVM) is an important classification method for a binary classification problem (Joachims, 2001). SVM maps a given set of n dimensional input vectors nonlinearly into a high dimensional feature space and separate the two classes of data with a maximum margin of hyperplane.

For the multi-class classification problem, a binary SVM is generated for each class c_i in general. Each SVM is trained for each binary classification problem. Given a new document d to be classified, each SVM estimates $P(c_i|d)$. The document is classified into the class c_i for which the corresponding $P(c_i|d)$ is highest. This reduction of a multi-class problem into m binary tasks is called a *one-versus-all* method (Joachims, 2001).

3. Results and Discussion

In this Section, we report our experimental results on the performance of Support Vector Machine (SVM). Experiments were performed on a 2.8GHz Pentium IV PC with 1GB of memory in Linux environment. Algorithms were coded with GNU C/C++. For SVM, we chose a linear SVM due to its popularity and fast training time compared to non-linear SVMs (e.g. polynomial, radial basis function, or sigmoid SVMs) in text classification (Joachims, 2001). It is important to select a good value of C, the amount of training error tolerated by the SVM, for the linear SVM. Among the possible values of $C \in \{0.05, 0.1, 0.5, 1.0, 5, 10, 1000\}$, we chose $C = 5$, since the linear SVM with $C = 5$ performed best on our datasets in terms of classification accuracy. We used the SVM $^{\text{multiclass}}$ [5] package by Joachims, which is an implementation of the multi-class SVM In this experiment we constructed three kinds of datasets (small, medium, and large datasets) to evaluate the performance of SVM algorithm.

Table 2: Number of citations with N classes

Dataset		#Citations with N classes				Total citations
		1	2	3	4	
Small	Training	8,741	29	0	0	8,799
	Training	74,988	605	0	0	76,198
	Test	31,961	356	0	0	32,673
Large	Training	129,364	14,780	418	6	160,202
	Test	55,634	6,234	188	2	68,674

[5] http://svmlight.joachims.org/

We used the holdout method to randomly divide each dataset into two parts: a training set and a test set. Table 1 lists the detailed information on the data sets used in this experiment and each data set contains 10 classes. After stemming and stop word removal, we obtained a vocabulary of 47,436 unique words for small dataset, a vocabulary of 217,872 distinct words for medium dataset, a vocabulary of 357,953 unique words for large dataset, respectively.

We investigated how many documents are contained in multiple relevant classes in our datasets. Table 2 lists the number of citations in each dataset and the number of documents with N classes ($1 \leq N \leq 4$).

To construct training and test data, we first surveyed how many GO terms are contained in PUBMED citations. For each GO term, we made a query statement, limiting the results to the Medical Subject Heading (MeSH) major topic field and to citations with abstracts in English (National Library of Medicine). After submitting all query statements to PUBMED, we found that a total number of 564 out of 17,593 GO terms found in PUBMED citations. Table 3 lists the top 10 most frequently occurring GO terms in PUBMED.

Table 3: Top 10 GO terms

GO ID	GO Term	#Citations
GO:0001469	protein	1,383,991
GO:0000417	cell	762,860
GO:0004433	peptide or protein amino-terminal blocking	681,065
GO:0004443	peptide or protein carboxyl-terminal blocking	681,065
GO:0005741	physiological process	239,942
GO:0003091	plasma protein	201,665
GO:0003382	nucleic acid	198,735
GO:0001124	behavior	193,769
GO:0004886	cell growth and/or maintenance	162,424
GO:0003067	peptide hormone	154,067

For evaluating the performance, we use the standard recall, precision, and F1 measure. Recall (r) is defined to be the ratio of correct predictions by the classification system divided by the total number of correct predictions. Precision (p) is the ratio of correct predictions by the classification system divided by the total number of the system's predictions. The F_1 measure combines recall and precision into an equally weighted single measure as follows:

$$F_1(r,p) = \frac{2rp}{r+p}$$

As a feature selection method, mutual information (or information gain) (Cover and Thomas, 1991) was used to select a total of 200, 600, and 1000 features that have the highest average mutual information with the class variable for each dataset.

Table 4 summarized the performance scores, precision (p), recall (r), and F_1 measures on three datasets for vocabulary sizes of 200, 600, and 1000 words. Compared to NB, SVM performs extremely well, except training times of classifiers (time: CPU seconds) (Table 5). We have carried out the long-waiting training times of 29190, 32071, 50065 CPU seconds. Despite this problem, SVM can be scaled up using grid computers of size 200 and above which are commonly operating in large research labs. Although PUBMED has about 2,000 new entries everyday, we will not retrain the whole data collection.

Table 4: Performance summary of SVM

Dataset		#Words								
		200			600			1000		
		p	r	F_1	p	r	F_1	p	r	F_1
Small	NB	89.27	89.51	89.39	91.75	91.90	91.83	91.70	91.83	91.77
	SVM	93.65	93.78	93.71	94.08	94.22	94.15	94.03	94.16	94.09
Medium	NB	80.83	81.24	81.03	85.52	86.16	85.54	87.28	87.97	87.76
	SVM	89.84	90.15	89.99	91.39	91.62	91.51	91.63	91.86	91.74
Large	NB	71.75	73.16	72.45	76.58	78.40	77.48	78.37	79.80	79.08
	SVM	81.73	81.74	81.74	83.62	83.70	83.66	83.90	83.97	83.93

Table 5: Train and test times of classifiers (time: CPU seconds)

Dataset		#Words: 200		#Words: 600		#Words: 1000	
		Train	Test	Train	Test	Train	Test
Small	NB	0.11	0.49	0.22	0.94	0.28	1.26
	SVM	29.01	0.02	33.16	0.04	35.90	0.03
Medium	NB	0.85	3.57	1.66	6.97	2.13	9.02
	SVM	4818.69	0.16	5115.62	0.28	5571.04	0.32
Large	NB	1.69	7.18	3.35	14.53	4.40	19.35
	SVM	29190.06	0.47	32071.2	0.55	50065.7	0.69

Table 5 shows the time to train and classify the NB and SVM algorithms for each dataset. The training time of linear SVM tends to increase dramatically with an increasing training set size and feature set size, although a linear SVM can be trained much faster than a nonlinear SVM (Joachims, 2001).

The results were executed on a 2.8GHz Pentium IV PC with 1GB of memory in Linux environment. If we scale up the environment to high performance computing (e.g., 200+ CPU's), we feel that SVM is a viable algorithm to implement the automated linking of PUBMED documents with GO terms. The reason that linear SVM is viable for parallelization is that the two mathematical operations of SVM can be parallelized:

A linear mapping of an input vector into a high dimensional feature space that is hidden from the input and output.

Construction of an optimal hyper-plane from features discovered in Step 1.

This hyper-plane is a decision surface that is constructed that separates members of different classes in such a way as to maximize the distance between them. The finding of hyper-plane is a convex optimization problem. The simplest solution is the gradient ascent approach that follows the steepest ascent path to the optimal solution. A more efficient way is to use the chunking decomposition algorithms. The basic idea of parallelism is derived from these two algorithms, which distribute the dataset to different processors and then aggregating results until convergence. The convergence criterion is the Karush-Kuhn-Tucker conditions. It is easy to develop that the parallel algorithm will always converge for distributed data sets.

References

Chakrabarti, S. (2000). Data mining for hypertext: A tutorial survey. *ACM SIGKDD Explorations* **1**, 1-11.

Cover, T. and Thomas, J. (1991). *Elements of Information Theory.* Wiley.

Hearst, M. A. (1999). *The Use of Categories and Clusters for Organizing Retrieval Results.* In *Natural Language Information Retrieval* (Edited by T. Strzalkowski), 333-374. Dordrecht, Kluwer Academic Publishers.

Joachims, T. (2001). *Learning to Classify Text using Support Vector Machines.* Kluwer Academic.

Kohonen, T. (1998). Self-organization of very large document collection: state of the art. *Proceedings of ICANN98, the 8-th International Conference on Artificial Neural Networks*, Skovde, Sweden.

Lewis, D. D. (1998). Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. *Proceedings of 10th European Conference on Machine Learning (ECML)*, 4-15. Chemnitz, Germany.

McCray, A. T., Browne, A. C., and Bodenreider, O. (2002). The lexical properties of the gene ontology (GO). *Proceeding of AMIA Annual Symposium*, 504-508.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Raychaudhuri, S., Chang, J. T., Sutphin, P. D., and Altman R. B. (2002). Associating genes with gene ontology codes using a maximum entropy analysis of biomedical literature. *Genome Research* **12**, 203-214.

Smith, T. C. and Cleary, J. G. (2003). Automatically linking MEDLINE abstracts to the gene ontology. *Proceedings of the ISMB 2003 BioLINK Text Data Mining*, Brisbane, Australia.

Smith, B., Williams J., and Schulze-Kremer, S. (2003). The ontology of the gene ontology. *Proceedings of the Annual Symposium of the American Medical Informatics Association*, 609-613.

Yandell, M. D. and Majoros, W. H. (2002). Genomics and natural language processing. *Nature Reviews Genetics* **3**, 601-610.

Received December 13, 2005; accepted February 9, 2005.

Su-Shing Chen
Computer and Information Science
and Engineering Department
University of Florida, Gainesville
Florida 32611, USA
suchen@cise.ufl.edu

Hyunki Kim
Computer and Information Science
and Engineering Department
University of Florida, Gainesville
Florida 32611, USA
hkk@etri.re.kr