Automatically linking MEDLINE abstracts to the Gene Ontology

Tony C. Smith
Department of Computer Science
University of Waikato
Hamilton, New Zealand
tcs@cs.waikato.ac.nz

John G. Cleary
Reel Two Ltd
9 Hardley Street Hamilton
New Zealand
jcleary@reeltwo.com

1 Introduction

Much has been written recently about the need for effective tools and methods for mining the wealth of information present in biomedical literature (Mack and Hehenberger, 2002; Blagosklonny and Pardee, 2001; Rindflesch et al., 2002)—the activity of conceptual biology. Keyword search engines operating over large electronic document stores (such as PubMed and the PNAS) offer some help, but there are fundamental obstacles that limit their effectiveness. In the first instance, there is no general consensus among scientists about the vernacular to be used when describing research about genes, proteins, drugs, diseases, tissues and therapies, making it very difficult to formulate a search query that retrieves the right documents. Secondly, finding relevant articles is just one aspect of the investigative process. A more fundamental goal is to establish links and relationships between facts existing in published literature in order to "validate current hypotheses or to generate new ones" (Barnes and Robertson, 2002)—something keyword search engines do little to support.

One promising solution is to bring biomedical literature into the structured organisation of the Gene Ontology (GO) (Consortium, 2000). A large number of genomic/proteomic databases (e.g. SwissProt, SGD, InterPro, FlyBase, etc) make use of GO in some way to link and unify expression data, organize genes and proteins into more or less coherent functional groups, and resolve some of the ambiguities in nomenclature, but little progress has been made towards exploiting GO directly with documents. For example, a substantial search effort made by the authors of this paper in mid-2002 found fewer than thirty thousand MEDLINE abstracts directly or indirectly linked to GO terms in public databases. The situation has improved greatly over the past year, such that a more recent search (completed in April 2003) uncovered about 120,000 MEDLINE abstracts linked to the Gene Ontology, but it will still take a very long time before all six million abstracts contained in the MEDLINE database[1] are associated with GO terms if the process continues to be done manually.

This paper describes the "Gene Ontology Knowledge Discovery System" (GO-KDS), a publicly available web application (www.go-kds.com) that uses machine learning techniques to automatically connect biomedical documents to terms from the Gene Ontology. General semantic models for each GO term are inferred from training documents gleaned from the references available in public gene/protein databases. The models are subsequently used to automatically classify all MEDLINE abstracts to appropriate GO terms.

2 GO-KDS: The Technology

Machine learning is widely used in bioinformatics research, with applications to various gene mining tasks—such as peptide identification, microarray and mass-spectral analyses, EST correction, and so forth. The basic idea of machine learning is to create computer programs that learn how to perform some task based upon observations about sample judgments made by human experts.

In the case of document classification, the expert gives the learning algorithm some number of documents deemed exemplars of a particular semantic class (and usually some number that are not). The algorithm identifies all salient features of the documents and weights those that are the best indicators for determining whether or not each document is an instance of the concept being learned. The result is a characteristic computer model that can subsequently be used to predict how likely it is that any future novel document also belongs in that semantic class.

[1] MEDLINE is often cited as having 12 million abstracts, but about half are actually retractions and corrections.

2.1 The training data

The training data for GO-KDS was obtained from those publicly available gene databases that include references to MEDLINE (or PubMed) documents and support direct or indirect associations to the Gene Ontology[2]. Approximately 26,500 training documents for nearly 3700 different GO terms were obtained in this way[3].

2.2 Text preprocessing

The features of a document that are most useful for predicting its class are primarily its words (though metadata can occasionally be useful). To make the words accessible for the machine learning algorithm, it is necessary to preprocess the input documents in a way that makes the general semantics of each document as conspicuous as possible in the token stream.

Empirical studies revealed that many general linguistic operations were unhelpful for GO-KDS. For example, several stemmers were trialed (including a Porters-style greedy stemmer and a rule-based inflectional lemmatizer) but failed to deliver general improvement in document classification tasks. Similalrly, bigrams and soft-parsed constituents made significant demands on system resources without delivering increased accuracy. Even a custom-built "chemical name" parser (i.e. a morphological analyzer that would, for example, parse a word like "proteoglycan" into "proteo" and "glycan") proved to be of little value. Ultimately, tokenization was restricted to the removal of function words (i.e. stop words), the expansion of Unicode abbreviations and symbols, and the parsing of complex gene names (e.g. a gene name such as "apo-H64Y/V68F" generates the features "apo-H64Y/V68F", "apo", "H64Y" and "V68F").

2.3 Learning Algorithm

In selecting appropriate machine learning technology there were a number of stringent constraints. First was the need for the underlying algorithm to be able to scale to large numbers of categories (around 3,500), training documents (about 30,000) and an even larger number of documents to be categorized (about 6 million). Within the set of training documents there are about a million distinct words that were used as features. Another issue was the extreme skewness of most of the categories - some contained as few as two positive training documents out of 30,000.

The traditional algorithms which have been used for this type of application are Näive Bayes (NB) and Support Vector Machines (SVM) (Mitchell, 1997). We experimented with using a number of different SVM implementations. On small subsets of the data SVM was able to achieve good accuracy. Unfortunately the large and uncertain memory requirements of this algorithm coupled with the super-linear dependence of its execution time on the number of training instances means that it was infeasible to use more than a few hundred training instances.

We also implemented a straightforward Näive Bayes algorithm but found that its accuracy was too low to be useful. This experience contradicts that reported elsewhere in the literature (Mitchell, 1997).

To get maximal accuracy both Näive Bayes and SVM require an initial pass to select a feature set. During our initial investigations this was felt to be too slow to be feasible. In retrospect we could probably have engineered it to work but given that the WCL system does not require it for good performance we have not revisited this issue.

Given this experience we decided to build our own underlying algorithm. It is loosely based on Näive Bayes in that it makes use of the same "bag of words" statistics that Näive Bayes does. That is, we only keep track of how many times each word has occurred in a document of each class. An advantage of this is that like NB we can easily make use of leave-one-out (LOO) predictions for evaluating our performance.

The LOO procedure is used when predicting documents taken from the training set. First the statistics for the document are subtracted from the underlying word counts, then the prediction is done and finally the statistics are restored. The time for this is roughly the same as for adding a document to the statistics so it is feasible to do it for all documents in the training set. This LOO prediction is thus not biased by overfitting, that is, it is a true reflection of how new previously unseen documents are likely to be predicted by the system.

2.3.1 WCL

The WCL algorithm assigns a score S to each document D as follows:

$$S = \sum_{w \in D} f(w)$$

where $f(w)$ is some function of the statistics for the word w. The final score S is used for comparatively ranking different documents within one class, later we will deal with the issue of actually computing probabilities of membership. Both NB and SVM fit within this framework. For SVM the values for $f(w)$ are computed by an iterative relaxation algorithm. NB is formulated this way by taking $f(w) = log(P(w|C))$ where $P(w|C)$ is an estimate of the probability of the word w given that the document is in category C.

In describing the actual formulation of $f(w)$ for WCL we will give a series of refinements. The first of

[2]The databases used were SwissProt, GenBank, FlyBase, GOA, Gramene Oryza, MGI, PomBase, RGD, SGD, TAIR, TIGR, WB, InterPro and AmiGO.

[3]The May 2003 search found over 110,000 MEDLINE documents associated with 4700 GO terms, but results from this new data are not yet available

these looks like an incorrect version of NB, $f(w) = log(P(C|w)) - log(P(C))$. ($P(C|w)$ is an estimate of the probability that the document is in category C given that the word w is in the document. $P(C)$ is an estimate of the probability that the document is in category C). The intuition here is that $f(w)$ is zero when $P(C|w)$ and $P(C)$ are the same, that is, w is completely uninformative. As is typical for such estimates we let the estimators $P(C|w) = n_{w,C} + 1/2/n_w + 1$ and $P(C) = n_C + 1/2/n + 1$ (n is the total number of documents, n_w is the number of documents that the word w appears in and $n_{w,C}$ is the number of documents that are in category C and which contain the word w.) Note that using these estimators $log(P(C|w)) = log(n_C + 1/2) - log(n_w + 1)$.

Estimators of the form $(n + \alpha)/(N + 1)$ are obtained by assuming a prior probability distribution of $p^{1-\alpha}$ and computing the Bayesian estimate of the expected posterior probability. However, because we are using the $log(P(C|w))$ in $f(w)$ what we should strictly do is estimate the expected logarithm of the probability rather than the probability itself. This gives an estimator of the form $\psi(n + \alpha) - \psi(N + 1)$ replacing log by psi. ($\psi(x)$ is the digamma function (Abramowitz and Stegun (Eds.), 1972)). Now we get $f(w) = \psi(n_{w,C} + 1/2) - \psi(n_w + 1) - \psi(n_C + 1/2) + \psi(n + 1)$.

This formulation gave a significant performance improvement over the simple logarithmic form. However, like NB and SVM we found that selecting a feature set of words was necessary in order to get the best performance. This is unsatisfying because adding more information (that is the statistics for words outside the feature set) should not degrade performance. One problem that was apparent was that words that occurred very seldom, say once or twice, could have high values for $f(w)$ and were contributing unduly to to the final scores. To reduce the contribution of such words we formulated a function $\sigma(w)$ which estimates the standard deviation of $f(w)$. Then the score can be reformulated as:

$$S = \sum_{w \in D} f(w)/\sigma(w)$$

The expected value of the standard deviation of the logarithm of the probabilities is $\sigma(w) = \psi_1(n_{w,C} + 1/2) - \psi_1(n_w + 1) - \psi_1(n_C + 1/2) + \psi_1(n + 1)$. ($\psi_1$ is the first polygamma function which is the derivative of ψ).

This formulation gave a further significant improvement to performance. However, another issue arose that the scores were dependent on the size of the documents. That is, a document with many words often gave a much larger score. This became an issue when some of the documents we were working with were short abstracts and others were full academic papers. To correct for this the score was normalized to allow for the length using:

$$N = \sum_{w \in D} 1/\sigma(w)$$

and then setting the final corrected score to be

$$R = S/N.$$

It is this formulation that was used in GO-KDS.

2.3.2 Calibration

The score obtained above only ranks documents within a particular category it makes no decision about actual membership in the category. In order to compare our results with other techniques such as SVM the facility was included to determine a threshold so that scores above the threshold were considered to be in the category. SVM does this automatically as part of its execution.

WCL does this by first collecting the LOO prediction score for each training document. This allows the number of incorrect decisions to be calculated for each possible setting of the threshold. The actual threshold is chosen as the *breakeven* point where the number of false positives equals the number of false negatives and where the precision and recall are the same.

As well as a threshold GO-KDS needs an estimate of what the probability of membership is given a score for a document. It uses these for providing feedback to users about the probability of membership (it does this crudely with a one to five star system) and also to avoid indexing large numbers of documents with low (less than 10%) probabilities of membership.

The probabilities are estimated using the same set of LOO scores as used for computing the threshold. An adaptive subdivision is done of the range of scores and a count made of the number of positives and negatives within each region. These counts are then used to estimate the true probabilities. Care is taken in this process to ensure that the probability assignments are a strictly increasing function of the scores.

2.3.3 Engineering

It was important to get a good fast underlying algorithm. WCL provides this by ensuring that memory and time usage are at worst linear in the size of the documents. Also it permits LOO predictions which allows fast and accurate evaluation of performance. That said, significant engineering was still required to get satisfactory performance. This centered around carefully sharing the tables of word counts between different category models and using sparse compact representations. The system was written in Java which gives ease of portability across different operating systems but which does require care to ensure that the object oriented nature of Java does not unduly slow down execution.

The final result of this is that using a single commodity Pentium IV processor with a gigabyte of RAM it is possible to build the models for GO-KDS and do an LOO evaluation of the training documents in less than an hour. The

complete indexing of the 6 million documents in MEDLINE takes 1.5 days using 5 commodity processors.

2.4 Classification accuracy

Measuring the perfomance of WCL against any sort of baseline is difficult because published results for a study of comparable scale using another classification scheme are not available. Perhaps the closest work is that done by Raychaudri *et al.* (Raychaudhuri et al., 2002), where a "maximum entropy" technique was employed to categorize 21 GO terms using training and test documents extracted from PubMed using handcrafted keyword queries. Their study reports that models trained on Medline documents published prior to 2001 achieved an accuracy of 72.8% when tested on documents published in 2001. To make the ocmparison, an attempt was made to recreate their sample corpus as best as possible, and experimentation with GO-KDS on the same 21 categories achieved an accuracy of 70.5% (at the precision-recall breakeven point). While these numbers are not directly comparable, they do indicate that the weighted confidence learner is delivering acceptable classification accuracy, and therefore that GO-KDS offers a practical way to connect vast amounts of biomedical literature to the gene ontology. The major bottle-neck to further improvements, both in terms of accuracy and coverage of the ontology, is obtaining more good quality training data. Our hope is that GO-KDS itself can be used to bootstrap this process allowing putative members of categories to be selected and then have humans check these suggestions.

3 Remarks

The growing need for effective text mining applications specifically for biologists is widely recognized, where "it is becoming increasingly more difficult to keep up with the avalanche of information flooding research journals" (Krauthammer et al., 2002). Keyword searching of electronic document stores is useful but limited by the fact that "synonyms abound in free text, and there are multiple ways of expressing the same idea ... The ambiguities in free text must be reconciled with the rigorous structure required by computers. This problem is unsolved and difficult" (Chang and Altman, 2002). Controlled vocabularies like MeSH "go some way to standardizing keyword searching ... However, MeSH does not provide the level of detail or sophistication needed to ensure precision and recall of relevant abstracts for the drug discovery scientist" (Barnes and Robertson, 2002). These facts have led many writers to speculate on the tremendous potential offered by structured ontologies as mechanisms to control the context of computational searches over published reports. "By capturing knowledge about a domain in a sharable and computationally accessible form, ontologies can provide defined, accessible and computable semantics about the domain knowledge they describe" (Lord et al., 2002).

GO-KDS uses text mining techniques to automatically connect research documents to ontology terms, thereby amplifying the potential of GO to elucidate the knowledge embedded within biomedical literature.

References

M. Abramowitz and I. A. Stegun (Eds.). 1972. Psi (digamma) function. In *CH. 6.3 in Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing*, pages 258–259. Dover, New York.

J. Barnes and J. Robertson. 2002. The use of ontologies in drug discovery. *Bioinformatics World*. www.bioinformaticsworld.info/biwaut02ontologies.html.

M. V. Blagosklonny and A. B. Pardee. 2001. Conceptual biology: unearthing the gems. *Nature*, 416:373.

J. T. Chang and R.B. Altman. 2002. Promises of text processing: natural language processing meets ai. Technical report smi-2002-0934, Stanford Medical Informatics, Stanford, California.

The Gene Ontology Consortium. 2000. Gene ontology: tool for the unification of biology. *Nature Genet.*, 25:25–29.

M. Krauthammer, Kra P., Iossifov I., Gomez S. M., Hripcsak G., V. Hatzivassiloglou, C. Friedman, and A. Rzhetsky. 2002. Of truth and pathways: chasing bits of information through myriads of articles. *Bioinformatics*, 18 Suppl 1:S249–S257.

P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble. 2002. Investigating semantic similarity measures across the gene ontology: the relationship between sequence and annotation. *Bioinformatics*. accepted for publication.

R. Mack and M. Hehenberger. 2002. Text-based knowledge discovery: search and mining of life-sciences documents. *Drug Discovery Today*, 7 Suppl.:589–598.

T. M. Mitchell. 1997. *Machine Learning*. McGraw Hill., New York.

S. Raychaudhuri, J. T. Chang, P. D. Sutphin, and R. B. Altman. 2002. Associating genes with gene ontology codes using a maximum entropy analysis of biomedical literature. *Genome Research*, 1:203–214.

T. C. Rindflesch, L. Tanabe, J. N. Weinstein, and L. Hunter. 2002. Edgar: Extraction of drugs, genes and relations from the biomedical literature. citeseer.nj.nec.com/277820.html.