GO for Gene Documents

Xin Ying Qiu
Management Sciences Department
Tippie College of Business
The University of Iowa

xin-qiu@uiowa.edu

Padmini Srinivasan
Management Sciences Department &
School of Library and Information Science
The University of Iowa

padmini-srinivasan@uiowa.edu

ABSTRACT

Annotating genes and their products with Gene Ontology codes is an important area of research. One approach for doing this is to use the information available about these genes in the biomedical literature. Our goal, based on this approach, is to develop automatic methods for annotation that could supplement the expensive manual annotation processes currently in place. Using a set of Support Vector Machines (SVM) classifiers we were able to achieve Fscores of 0.48, 0.4 and 0.32 for codes of the molecular function, cellular component and biological process GO hierarchies respectively. We explore thresholding of SVM scores, the relationship of performance to hierarchy level and to the number of positives in the training sets. We find that hierarchy level is important especially for the molecular function and biological process hierarchies. We find that the cellular component hierarchy stands apart from the other two in many respects. This may be due to fundamental differences in link semantics. This research also exploits the hierarchical structures by defining and testing a relaxed criteria for classification correctness.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.7 [**Document and Text Processing**]: Miscellaneous

General Terms

Experimentation, Performance

Keywords

Automatic document annotation, Gene Ontology, Hierarchy structures

1. INTRODUCTION

Annotating genes and their products with Gene Ontology codes is an important area of research. One approach for doing this is to use the information available about these genes

in the biomedical literature. Our goal, based on this approach, is to develop automatic methods for annotation that could supplement the expensive manual annotation processes currently in place.

The importance of this GO annotation problem and the value of computational methods to solve for it are well recognized. In the 2004 BioCreAtIve challenge, a set of tasks were designed to assess the performance of current systems in the area of supporting GO annotations for specific proteins. In particular, the second task of identifying specific text passages that provide the evidence for annotation resembles most the manual process of GO annotation[7]. The participating systems showed a variety of approaches (from heuristics to Support Vector Machines classification) exploring different levels in text analysis (such as sentences or paragraphs)[2]. In Rice et al.[12], Support Vector Machines (SVM) classification was applied to the relevant documents for each GO code. Features from the documents were selected and conflated as sets of synonymous terms. Their methods worked better when a substantial set of relevant documents were available. In Ray et al.[11], statistical methods were first applied to identify n-gram informative terms from the relevant documents of each GO term. These term models provided hypothesized annotation models which could be applied to the test documents. In Chiang et al.[5], a hybrid method that combined sentence level classification and pattern matching seemed to achieve higher precision with fewer true positive documents.

In some of these previous studies, the hierarchy was explored but to a limited extent. This was done primarily to add information to the classification models. When working on GO annotation one may certainly draw from the general hierarchical text classification literature (e.g. [4], [6], [18]). We may also learn from hierarchical efforts with MeSH[14]. However GO may have special characteristics that could be exploited beneficially. Or there may be properties that must be considered by automatic annotation systems in order to be effective.

Our goal in this research is to gain a better understanding of the GO annotation problem using Support Vector Machines classification algorithms. We will study several open issues in the GO context. One is the effect of the hierarchical level on performance. Another is the effect of skewed distributions where the negative examples tend to overwhelm the positives in the training data. We will also study differences between hierarchies built predominantly upon is_a relationships and those that significantly include $part\_of$ relationships as well. Although both are asymmetric and transitive,

their semantics are very different. Looking beyond achieving good performance, our aim in this research is to contribute to an understanding of the problem itself. The annotation of genes and their products is an important contribution to developments in bioinformatics. As new genes are discovered and as new functions of genes are identified, these annotations serve as key mechanisms for organizing and providing access to the accumulated knowledge.

2. DATA SOURCES AND APPROACH

2.1 Gene Ontology

Gene Ontology (GO)[1] provides a structured vocabulary that is used to annotate gene products in order to succinctly indicate their molecular functions, biological processes, and cellular components[1]. Although different subsets of GO may be used to annotate different species, the intent is to provide a common annotation infrastructure. Molecular function describes activities performed by individual gene products or complexes of gene products. Examples of molecular functions are *arbutin transporter activity* and *retinoic acid receptor binding*. A biological process is made of several steps accomplished by sequences of molecular functions. Examples include *lipoprotein transport* and *phage assembly*. Cellular components are for example, the *nucleus*, *NADPH oxidase complex*, and *chromosome*. There are three hierarchies in GO corresponding to these major dimensions. Each hierarchy is a directed acyclic graph (DAG). The molecular function hierarchy almost completely consists of is_a links. About a fifth of the links in the biological process hierarchy represent part_of links and the rest are is_a links. The cellular component hierarchy is about evenly balanced between the two types of links.

2.2 Annotations

We began with the August 2005 download of LocusLink and extracted the entries for Homo Sapiens limited to those with locus type *gene with protein product, function known or inferred.*.

There are $77,759$ annotation entries for $16,630$ locus ids. Considering only annotations that used documents for evidence we have $29,501$ entries. These entries are then limited to those having TAS (Traceable Author Statement) or IDA (Inferred from Direct Assay) as evidence types yielding $20,869$ entries[2]. These entries are composed of $9,577$ annotations for biological processes (BP) $5,195$ annotations for cellular components (CC) and $6,097$ for molecular function (MF). Together these $20,869$ annotations reference $8,744$ unique documents.

We looked at the distribution of the GO codes in our dataset in terms of the number of documents associated with each. The range is 1 to 333 for MF, 1 to 789 for CC and 1 to 579 for BP.

Limiting ourselves to only those codes that had at least 5 (unique) documents associated, we get 283 unique codes for BP, 93 for CC and 214 for MF. We used 5 as the threshold given the 5 times cross validation design for our experiments. Thus we wish to ensure that each code had at least 1 evidence document in each split. Interestingly some code

- pmid combinations occur more than once. This happens when the same document offers two different kinds of evidence, say TAS as also IDA, for annotation. Limiting these combinations to the unique occurrences gives us $7,200$ annotations for BP, $4,391$ for CC and $3,877$ for MF. These data were used in our experiments.

The data for each hierarchy was randomly split into 5 splits such that each code appears in each split with near equal numbers of evidence documents. The overall cross validation strategy is to iteratively take 4 splits as training data and test the trained model on the remaining fifth split. As an example for split1, we take splits 2 - 5 as training data and 1 as testing. This ensures that there are at least 4 relevant documents for a code in the training side and at least 1 in the test side.

2.3 Document Representation

In information retrieval research, the most widely used document representation method is the "bag of words" approach where all the terms are used to form a vector representation. Functional or connective words are considered as stop words and are generally removed since they are assumed to have no information content. The terms could be weighted for example, with TF×IDF weights or boolean weights. Alternative methods of defining terms have been explored, but with little significant improvement for text classification performance. Recent research by Moschitti and Basili[10] suggests that the elementary textual representation based on words applied to SVMs models is very effective in text classification. More complex linguistic features such as part-of-speech information and word senses did not contribute to the predictive accuracy of SVMs.

For this research, we use vector representation for documents produced using the SMART system[15] with stemmed terms after removing stop words. The "atc"[17] construction of TF×IDF weighting scheme were applied to the terms. This representation has worked well in our previous research ([9]). We used the title, abstract, RN and MeSH fields of the MEDLINE records.

2.4 Overall Approach

Genes (or more strictly their products) are annotated with GO codes. Our interest is in predicting annotations from the literature, specifically from MEDLINE records. This is in contrast to other annotation methods such as the ones involving sequence homology and protein domain analysis (e.g. [19]). We approach the MEDLINE based annotation problem in three phases. In the first phase we find documents that are relevant to the gene. In the second phase we determine which codes should be assigned to each document. In the third phase we decide which codes should be assigned to a gene/gene product based on its classified documents. In recently completed work we studied phase 1, the problem of retrieving MEDLINE records for genes[16]. In it we consider the special challenges of dealing with gene name and symbol ambiguity. In this research we focus mainly on phase 2. That is, given a document (relevant for a gene or a gene product) we ask: what GO codes should be assigned to it? We also close this paper with preliminary results for phase 3 using a very simple strategy. Specifically a gene is assigned a code if it is assigned to any of its relevant documents. More sophisticated strategies for phase 3 are left to future research.

---

[1]Downloaded on May 16 2006 from the GO Consortium: http://www.geneontology.org

[2]http://www.geneontology.org/GO.evidence.shtml.

The document annotation or classification problem of phase two is interesting in that the codes themselves are structured hierarchically. Similar hierarchical classification problems have been addressed ([4], [6], [18]) including by our own group ([13], [14]). A key aspect in GO based research is that we have three hierarchies with different properties. Moreover with GO, document classification is not the end point but a step toward the goal which is gene/gene product annotation (i.e., phase 3).

We adopt a classifier-based machine learning approach using the open source software SVM Light[3]. In all experiments parameters are set at their default values. The positive instances for a GO code are those records associated with it in the LocusLink dataset. The negative instances are records assigned to all the other GO codes.

We present a sequence of experiments within the Support Vector Machines classifier framework. These also explore the effect of hierarchy level and number of positives available for each code during model building. We also explore a more relaxed definition of classification correctness. Our overall aim is to contribute a better understanding of phase 2 of the GO annotation problem.

3. CODE SPECIFIC SVM CLASSIFIERS

Support Vector Machines were designed for binary or 2 class classification problems. A common solution, adopted here, is to transform an N class problem into N binary problems. Thus we build a distinct classifier for each code (class) where the classifier decides whether a document belongs to the code's class or not. The hierarchy within each GO dimension is not used at this point. The only connection among the codes is that they share a common dataset, albeit with different positive and negative instances. Unfortunately, this approach yields extremely poor results as shown in table 1. We found that most of the scores calculated by SVM are negative, mainly due to the highly skewed nature of the training data for most codes. As observed by several others this problem may be fixed with judicious thresholding[3]. So in the next experiment we calculated an optimal threshold from the training data for the SVM scores.

Hierarchy	Recall	Precision	Fscore
MF	0.0419	0.0944	0.052
CC	0.0599	0.1461	0.0764
BP	0.0234	0.064	0.0398

Table 1: Results: Single Classifier for each GO Code

4. SVM SCORE THRESHOLDS

Our goal is to determine a single threshold score for each hierarchy, such that documents with scores assigned by the SVM classifier above this threshold are declared positive. We select the best threshold from the training data for each split. In particular, we take the training dataset of a split and divide it into 4 parts. (We call these 'folds' in order to maintain a distinction from the higher level 'splits'). Cross validation over these four folds is done to generate a single best threshold which is then applied to the test side of the

split. The single best threshold was the average of the best thresholds in the four folds[3].

Results are presented in table 2. The table shows for each hierarchy, the threshold selected for each split as well as the recall, precision and Fscore values achieved on both the training and test sets. Averages across the splits are also provided. First we observe that the thresholds selected fall within a small range from -0.87 to -0.82 across all hierarchies. Molecular function has the smallest spread of threshold values (-0.85 to -0.84). We also observe that molecular function offers a relatively easier problem compared to cellular component with biological process being the hardest to solve. Finally, the test set scores are actually better than the training set scores indicating that we have successfully avoided over training our models in each case as these are able to generalize to the unseen test cases. Thus we see that setting the thresholds appropriately for these SVM classifiers offers enormous benefits in performance (when compared to the results in table 1).

5. CODE SPECIFIC THRESHOLDS

In the previous experiment a single threshold score was set for each hierarchy. In this experiment thresholds are set specific to individual GO codes. This strategy is reasonable to explore as it may indeed be that although the averages fall within a small range, the optimal threshold varies considerably across the codes. The overall structure of the experiment is the same as in the previous experiment. Code specific thresholds are set using a 4-fold cross validation experiment on each training set. The selected threshold is the average of the best threshold for the code across the 4 folds.

Results are presented in table 3. Interestingly, this time the Fscores achieved on the training runs are considerably higher than the Fscores achieved in the test runs of the single threshold experiment (compare with table 2). However, the penalty is clearly paid on the test side, indicating that this code specific strategy over-trains and fails to generalize effectively on new data. The one exception is in the case of CC where the Fscores are about the same in both cases. However, performance for MF and BP drop significantly by 10.4% and 17.5% respectively. Thus a single threshold over all codes of a hierarchy is superior to code specific thresholding. We also find a similar pattern with the previous experiment in that molecular function is easier to work with than cellular component which in turn is less challenging than biological process.

6. ANALYSIS OF RESULTS

We now analyze the best results obtained thus far which is obtained using a single threshold score for all codes of a given hierarchy. Our goal is to obtain further insights into factors influencing the results.

6.1 Recall versus Precision

It is well understood that the same Fscore may be obtained from different combinations of recall and precision. In this regard a key point to note from table 2 (and table 3) is that recall is always considerably higher than precision. Although recall could also be improved, our results indicate that the more serious problem for us lies in the context of precision. Although in general we are making the correct decisions the problem is we are making too many false pos-

Hierarchy	Split	Threshold	Training			Testing		
			Recall	Precision	Fscore	Recall	Precision	Fscore
MF	1	-0.84	0.5624	0.4136	0.4504	0.5992	0.4258	0.4684
MF	2	-0.86	0.5923	0.3835	0.4390	0.6775	0.4073	0.4769
MF	3	-0.86	0.5954	0.3734	0.4328	0.6817	0.3874	0.4684
MF	4	-0.84	0.5713	0.4046	0.449	0.6857	0.4487	0.5134
MF	5	-0.85	0.5921	0.4076	0.4541	0.6772	0.3945	0.4727
MF	Average	na	0.5827	0.3965	0.4451	0.6643	0.4128	0.48
CC	1	-0.82	0.4799	0.3185	0.3627	0.5301	0.3531	0.3986
CC	2	-0.82	0.4823	0.3214	0.3665	0.5359	0.3516	0.4006
CC	3	-0.86	0.5287	0.2976	0.3590	0.6553	0.3895	0.4571
CC	4	-0.85	0.5122	0.2997	0.3571	0.5703	0.2976	0.3715
CC	5	-0.85	0.5222	0.315	0.3714	0.599	0.29	0.3767
CC	Average	na	0.5051	0.3104	0.3633	0.5781	0.3364	0.4009
BP	1	-0.87	0.4304	0.2378	0.2847	0.4722	0.2585	0.3079
BP	2	-0.87	0.4377	0.2442	0.2908	0.5259	0.2713	0.3362
BP	3	-0.85	0.4019	0.2615	0.2948	0.4908	0.2884	0.3392
BP	4	-0.84	0.3706	0.2556	0.2794	0.4854	0.2966	0.3484
BP	5	-0.87	0.4519	0.2600	0.3069	0.4608	0.2220	0.2791
BP	Average	na	0.4185	0.2518	0.2913	0.4870	0.2674	0.3222

Table 2: Results: Using a Common SVM Score Threshold

Hierarchy	Split	Training	Testing		
		Fscore	Recall	Precision	Fscore
MF	1	0.6221	0.4499	0.3989	0.3852
MF	2	0.615	0.5364	0.4402	0.44351
MF	3	0.6128	0.5295	0.3892	0.4133
MF	4	0.6298	0.5793	0.4394	0.452
MF	5	0.6371	0.5467	0.4264	0.4451
MF	average	0.6234	0.5284	0.4188	0.4278
CC	1	0.5541	0.4679	0.3774	0.3842
CC	2	0.5052	0.5029	0.3435	0.3626
CC	3	0.5131	0.5632	0.3806	0.4239
CC	4	0.5554	0.5134	0.3273	0.3727
CC	5	0.5796	0.5148	0.3875	0.4201
CC	average	0.5415	0.5125	0.3632	0.3927
BP	1	0.4469	0.3994	0.2463	0.2554
BP	2	0.4472	0.4017	0.2727	0.2793
BP	3	0.4378	0.3951	0.2531	0.2589
BP	4	0.4248	0.4309	0.2654	0.2804
BP	5	0.4518	0.3710	0.2434	0.2543
BP	average	0.4417	0.3996	0.2562	0.2657

Table 3: Results: Using Dynamic Thresholds for SVM Scores

itive declarations. In other words we need to tighten the constraints and apply some filtering criteria on the positive decisions declared. This angle will be pursued in future research.

6.2 Hierarchical Level & Performance

Table 4 presents performance achieved for each level of the hierarchies. Note that levels increase with the depth of the tree. Thus more specific codes have higher level numbers. The table identifies the number of codes at each level as well as the average scores. For molecular function, ignoring level 1 which has very few codes, we find that levels 2 and 3 are the most challenging. The remaining MF levels achieve Fscore in the range of 0.4728 to 0.6667. However with the cellular component hierarchy we have Fscore decreasing as the level increases (barring level 1 which has only 1 code). Finally with biological process, after level 2, we observe somewhat stable performance between levels 3 and 6 (0.31 - 0.32 Fscore). Higher levels, especially level 7, show better performance.

It seems that with MF and BP hierarchies the difficult decisions are closer to the upper levels. This is contrary to common intuition which suggests that classifying into more general categories (such as animal or plant) should be easier than classifying into more specific categories (such as hawk or eagle). CC is different in that the decisions become more challenging as we descend the hierarchy. The difference between MF and BP on the one hand and CC on the other could be because of differences in the underlying semantics of the links. As mentioned before CC links are about evenly split between is_a and part_of whereas BP links are about 75% made of is_a links while MF is almost exclusively is_a. These performance differences observed across the levels of the hierarchies have important implications in the design of automated annotation systems for GO.

Hier.	Level	# of Codes	Scores		
			Recall	Precision	FScore
MF	1	4	0.3176	0.1786	0.2205
MF	2	26	0.4846	0.2666	0.3176
MF	3	41	0.5261	0.3145	0.3695
MF	4	50	0.6780	0.4449	0.5066
MF	5	57	0.7799	0.4936	0.5732
MF	6	17	0.8937	0.5548	0.6505
MF	7	11	0.6961	0.3876	0.4728
MF	8	4	0.675	0.475	0.5233
MF	9	2	0.8	0.6	0.6667
CC	1	1	0.3171	0.2235	0.2537
CC	2	20	0.6476	0.4017	0.4675
CC	3	25	0.6062	0.349	0.4089
CC	4	26	0.5547	0.306	0.3741
CC	5	14	0.5502	0.2832	0.3622
CC	6	6	0.3955	0.2717	0.3135
CC	7	1	1	0.7917	0.8667
BP	1	3	0.1354	0.0481	0.0704
BP	2	10	0.3327	0.1767	0.2174
BP	3	34	0.5164	0.2517	0.3179
BP	4	54	0.4849	0.2563	0.3119
BP	5	49	0.4681	0.2516	0.3093
BP	6	52	0.4555	0.2734	0.3139
BP	7	51	0.5863	0.3251	0.3921
BP	8	21	0.4677	0.2834	0.3301
BP	9	8	0.4698	0.2840	0.3316

Table 4: Performance by Level

6.3 Number of Positives for Training & Performance

Table 5 presents average scores for different ranges of number of positive examples in the training sets. Intuitively we expect less skewed training data to provide better results as we are using supervised SVM classifiers. Interestingly we observe that this does not necessarily hold. For example with molecular function higher numbers of true positives do not necessarily yield better Fscores. Limiting our attention to only those ranges with at least 10 codes, we find for example, having more than 150 examples is significantly worse than having just 16 to 20 positive examples. Observe that as the size of the training set size is the same for each code, having fewer positives implies that there are more negatives in the sample. With the cellular component hierarchy we restrict our attention to only the first 2 rows as the other cells have too few codes in them. Again we see that fewer examples yield better results. With the BP hierarchy we again see a similar tendency for performance to drop with increasing numbers of positive examples. The exception is the first row which has significantly lower Fscore than the next few ranges. These observations are interesting especially because they are counter to the generally accepted notion that with a supervised approach we may expect better results with more positive data.

Training size	# codes	MF-Fscore	# codes	CC-Fscore	# codes	BP-Fscore
5	2	0.25	34	0.4067	128	0.2695
6-10	3	0.0833	25	0.3650	65	0.3875
11-15	9	0.4373	7	0.4528	22	0.3716
16-20	37	0.5645	4	0.4550	15	0.3306
21-25	39	0.544	3	0.4762	9	0.2588
26-30	31	0.5566	4	0.3687	4	0.3007
31-35	6	0.4663	3	0.5651	8	0.3566
36-40	7	0.5275	0	0	6	0.3579
41-45	10	0.4124	1	0.2009	5	0.3484
46-50	11	0.4276	1	0.2861	2	0.2553
51-75	18	0.3912	2	0.3430	12	0.3060
76-100	12	0.3936	1	0.2681	6	0.2726
101-125	5	0.4273	2	0.4089	0	0
126-150	4	0.4767	2	0.3226	0	0
151-last	20	0.3511	4	0.4586	1	0.2822

Table 5: Performance by Number of Positives for Training

6.4 Correlations between Level and Number of Positives for Training

Taking this analysis the next logical step forward we explore the relationship between level, positive set size and performance for each code. Table 6 presents the computed correlations.

We find a moderate and significant negative correlation between level and size in the case of MF and BP but interestingly not in the case of CC. So with MF and BP more specific codes tend to have fewer positives in the training data but this is not the case with CC. There is also a moderate and significant positive correlation between level and FScore in the case of MF and BP but again not for CC. That is we tend to get better Fscores with more specific codes in MF and BP hierarchies but not so with CC. Thus with MF and BP we need to pay closer attention to the higher level codes. Once again our efforts indicate that CC is a hierarchy that might require classification methods that are different from those that are appropriate for MF and BP. Again this may be due to the underlying differences in link semantics.

Hier	Level vs Size	Level vs FScore	Size vs FScore
MF	-0.2705*	0.3361*	-0.1146
CC	-0.0123	-0.1051	0.0904
BP	-0.2155*	0.1622*	-0.0191

Table 6: Correlations. * - significant (0.01 significance level)

A second observation may be made from the correlations between performance and the other two variables. Specifically, level is far more important than the number of positives available for training, at least in the case of MF and BP. Thus in order to seek improvements in performance it would be prudent to develop methods capable of exploiting the level information for the GO codes. Size of training set on the other hand does not correlate with performance. As mentioned before this is a surprising observation given the commonly accepted notion that larger amounts of (positive) training data tend to yield better performance scores.

7. LEVEL SPECIFIC THRESHOLDS

To explore the effect of level further we adopt a simple strategy of setting the threshold by level. Table 7 shows the effect of this strategy for the MF and BP hierarchies, focussing only on levels 2 and 3. We do not apply this strategy to CC as there was no correlation between level and performance for this hierarchy. Also we consider only levels 2 and 3 as level 1 has too few codes and these are the levels where we seek improvements.

Interestingly, we find improvements at level 2 for both MF and BP (+7.4% and +4.6% improvements in Fscore respectively). However, the strategy does not work for level 3 in both cases. We will consider a different approach in future research, one that involves including examples from the neighborhood of the code. This could optionally include weighting by distance to the code.

Hier.	Split	Level	Original Fscore	Threshold	Final Fscore
MF	1	2	0.3299	-0.8	0.3665
MF	2	2	0.2782	-0.83	0.2973
MF	3	2	0.3298	-0.78	0.373
MF	4	2	0.3484	-0.81	0.373
MF	5	2	0.3016	-0.78	0.3263
MF	avg	2	0.3176	na	0.341 (+7.4%)
MF	1	3	0.3347	-0.87	0.3063
MF	2	3	0.3178	-0.84	0.3301
MF	3	3	0.4243	-0.88	0.3760
MF	4	3	0.4263	-0.87	0.3823
MF	5	3	0.3444	-0.86	0.3464
MF	avg	3	0.3695	na	0.3482 (-5.8%)
BP	1	2	0.2542	-0.87	0.2542
BP	2	2	0.2951	-0.89	0.2989
BP	3	2	0.2261	-0.89	0.2027
BP	4	2	0.1609	-0.87	0.2319
BP	5	2	0.1507	-0.88	0.1494
BP	avg	2	0.2174	na	0.2274 (+4.6%)
BP	1	3	0.2916	-0.86	0.3020
BP	2	3	0.3145	-0.83	0.3455
BP	3	3	0.3496	-0.82	0.3030
BP	4	3	0.3128	-0.83	0.3164
BP	5	3	0.3209	-0.83	0.3529
BP	avg	3	0.3179	na	0.324 (+1.9%)

Table 7: Performance: Level Specific Thresholds

8. RELAXING THE CORRECTNESS CRITERIA

Thus far we have not utilized the hierarchical structure in any way. There are at least two major directions in which the hierarchy may be utilized. One is where the hierarchy is used somehow during model building. For example, a node's training data may be augmented with training data from its neighbors ([11]). Alternatively, a top down approach for model building may be employed, with examples that filter through higher level nodes participating in lower level de-cisions ([4]). Many variations on these themes have been explored in the general machine learning literature. In this research we explore a second direction that has recently attracted the attention of researchers, especially in the context of bioinformatics problems (e.g. [8]). Specifically, we use the hierarchy to relax the criteria for correctness of a classification decision during evaluation. Essentially we assume that when a document is assigned a GO code it is implicitly assigned the ancestor GO codes as well. This is reasonable since the GO hierarchies encode is_a and part_of semantics along the parent - child links and these are transitive relationships. With this assumption we relax the calculation of recall and precision and therefore also of FScore as follows.

Recall = A/B where B is as usual the number of known correct code - pmid pairs in the dataset. The relaxation is applied to the calculation of A.

Consider a code - pmid pair (C - P) which is known to be correct. If our classifiers assign code C to P then A is increased by 1. Otherwise if our classifiers assign a code C' to P where C' is an ancestor of C then again A is increased by 1.

Precision = E/F where F is as usual the number of positive decisions declared by the classifiers. The relaxation is applied to the calculation of E.

Consider a code - pmid pair (C - P) which is declared a positive by our classifiers. If code C is correctly assigned to P then E is increased by 1. Otherwise if there exists a code C" which is known to be assigned to P where C is an ancestor of C" then E is increased by 1.

Note that our relaxed evaluation accepts as correct those decisions that are more general than the correct code and not those decisions that are more specific than the correct code. Thus if the target code is *glucoside transport*, we will accept as correct classification with the higher level (general) *carbohydrate transport* code but not classification with the lower level (specific) *alpha-glucoside transport* or *beta-glucoside transport* codes.

The definition of 'ancestor' can of course be varied depending upon how far up the tree one considers. This is formalized by ANCESTOR_LEVEL, a parameter that can be varied systematically. For example, when set to 1 ancestors are limited to parents. Table 8 presents our results using this relaxed evaluation scheme with ANCESTOR_LEVEL varying from 1 to 5. Unfortunately the results indicate that we do not achieve improvements in FScore even when we consider ancestors 5 levels up the hierarchies. But all is not lost as we see next!

Table 9 takes a different perspective on assessing performance within the context of this experiment. Note first that thus far results have been obtained from *averages of scores for each GO code*. To explain we have 5 splits in our experiment design (see section 2), and each GO code appears in each split with roughly equal number of positive examples. Within a split we first calculate FScore for each code and then average these FScores. Tables 3 and 4 show such averages for each split as also the global average. This approach for evaluation reflects a 'code' perspective with all codes being considered equally important. A different way to summarize performance is to consider each code - pmid combination as an independent decision that has to be made. Each combination needs to be declared as positive or negative by our classifiers. Thus given N codes and M pmids, $N \times M$ decisions are to be made. Averages may then be com-

puted across the set of decisions in a split. In table 9 results are presented from this perspective of individual decisions.

Observe first that we have new baselines identified for each hierarchy. Note also that from the decision perspective, CC is the easier hierarchy followed by MF and then BP. When compared to these baselines we find steady improvements as the definition of ancestor changes. Using ancestors up to 3 levels above gives improvements of 7.2%, 7.6% and 4.5% for MF, CC and BP respectively. With level 5 we have 7.4%, 8.8% and 6.1% respectively. These improvements indicate that, from a decision perspective, we perform better if we accept decisions that are approximately in the correct vicinity of the target code.

Is the decision perspective useful? The answer is yes. Averaging by the code (as done in the previous experiments) tells us which codes are more challenging than others. While designing annotation systems, we need to know code level differences that may lead to tailored strategies. For example the classifier system may differ by code level in the hierarchies. So the "code perspective" is certainly important. However, the decision perspective is more indicative of performance in terms of our end goal - annotation at the gene product level. The decision perspective implies that each annotation decision, irrespective of code, is equally important.

	ANC_LEVEL	Recall	Precision	Fscore
MF	baseline	0.6643	0.4128	0.4800
MF	1	0.6643	0.419	0.4847
MF	2	0.6650	0.4229	0.4880
MF	3	0.6650	0.4243	0.4888
MF	4	0.6650	0.4245	0.4890
MF	5	0.6650	0.4245	0.4880
CC	baseline	0.5781	0.3364	0.4009
CC	1	0.5781	0.3471	0.4082
CC	2	0.5784	0.3509	0.4113
CC	3	0.5784	0.3536	0.4132
CC	4	0.5784	0.3540	0.4136
BP	baseline	0.4870	0.2674	0.3222
BP	1	0.4887	0.2724	0.3265
BP	2	0.4887	0.2746	0.3285
BP	3	0.4890	0.2773	0.3301
BP	4	0.4890	0.2776	0.3305
BP	5	0.4890	0.2778	0.3306

Table 8: Performance: Common SVM Score Threshold Runs with Relaxed Correctness Criteria (Code Perspective)

Finally, we consider the annotation of the gene/gene product (i.e., the locus id) itself. We test a simple strategy of annotating a gene with a code if the code is assigned by our system of classifiers to a document that is relevant to the gene. Using this strategy we obtain for MF an Fscore of 0.31 (recall = 0.35 and precision = 0.28), for CC an Fscore of 0.36 (recall = 0.47 and precision = 0.29) and an Fscore of 0.22 for BP (recall = 0.26 and precision = 0.191). These scores are on the low side indicating that on the whole the problem of annotation is hard and one that offers many challenges. We observe that the order of difficulty for the hierarchies at the gene product annotation level has CC being easier than MF and then BP. This parallels the order observed with the

	ANC_LEVEL	Recall	Precision	Fscore
MF	baseline	0.6639	0.3076	0.4100
MF	1	0.6639	0.3195	0.4309
MF	2	0.6652	0.326	0.4370
MF	3	0.6652	0.3288	0.4396
MF	4	0.6652	0.3296	0.4403
MF	5	0.6652	0.3297	0.4404
CC	baseline	0.7442	0.3163	0.4432
CC	1	0.7442	0.3321	0.4586
CC	2	0.7458	0.3512	0.4769
CC	3	0.74583	0.3551	0.4804
CC	4	0.7458	0.357	0.4822
CC	5	0.7458	0.3572	0.4823
BP	baseline	0.5578	0.2286	0.3236
BP	1	0.5583	0.2360	0.3311
BP	2	0.5583	0.2432	0.3381
BP	3	0.5586	0.2466	0.3415
BP	4	0.5586	0.2482	0.3430
BP	5	0.5586	0.2485	0.3433

Table 9: Performance: Common SVM Score Threshold Runs with Relaxed Correctness Criteria (Decision Perspective)

decision perspective (see table 9). We view these phase 3 (of the gene annotation problem, see section 2.3) results as preliminary. Our focus in this paper is on gaining a better understanding of phase 2 which is document classification with GO codes.

9. CONCLUSIONS

We presented a series of experiments designed to explore the value of Support Vector Machine based classifiers for assigning Gene Ontology codes to MEDLINE documents. We find that by using thresholds selected for each hierarchy Fscores of 0.48, 0.4 and 0.32 are obtained for the MF, CC and BP hierarchies respectively. This is with a system of SVM classifiers that do not yet capitalize on the hierarchical organization of the codes. Interestingly, threshold selection at the individual code level (as opposed to the full hierarchy) decreases performance due to over training. We explored performance by level and by the number of positives in the training set. The former appears more important especially for MF and BP. CC in general differs from the other two hierarchies. This may be due to differences in link semantics as almost 50% of links are part_of in CC. In contrast, only a fifth of the links in BP are part_of and there is only 1 such link in MF. Setting level specific thresholds for the second highest level of MF and BP lead to appreciable improvements in Fscore. But this was not the case for level 3. Finally we explored a more relaxed evaluation criteria where classification with a more general code compared to the target code is considered correct. This yielded appreciable improvements when a decision perspective was taken during evaluation.

From this study we conclude that the hierarchies are different. Also hierarchical level is important. Counter to common intuition more general codes in MF and BP are actually more challenging for classification. Also counter to common intuition it is not necessarily the case that having more positives in our training data yields better performance.

There are several other ways in which we will exploit the hierarchical structure in future work. For example, we plan to try an ensemble of classifiers where ensembles are defined through the hierarchy. We also plan to focus more on the codes that have extremely few positive examples (1 to 4). In this study we employed (global) feature weighting in lieu of feature selection. In future work we will explore feature selection, both global and local (code specific) strategies, more directly. Finally, we plan on exploring other strategies for phase 3 of the annotation problem which is to determine the codes for a gene/gene product after these codes have been assigned to their relevant documents. The current study has given us a better understanding of the problem of classifying documents with GO codes and prepares us for future work in this direction.

Acknowledgments

10. REFERENCES

[1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[2] C. Blaschke, E. A. Leon, M. Krallinger, and A. Valencia. Evaluation of biocreative assessment of task 2. *BMC Bioinformatics*, 6(Suppl 1)(S16):291–301, May 2005.

[3] J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mlade. Training text classifiers with svm on very few positive examples. *Microsoft Corporation Technical Report, MSR-TR-2003-34*, 2003.

[4] S. Charkrabarti, B. Dom, R. Agrawal, and P. Raghavan. Using taxonomy, discriminants, and signatures for navigating in text databases. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1997.

[5] J.-H. Chiang and H.-C. Yu. Extracting functional annotations of proteins based on hybrid text mining approaches. In *Proceedings of BioCreAtIvE Challenge Evaluation Workshop 2004*, 2004.

[6] S. Dumais and H. Chen. Hierarchical classification of web content. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR) 2000*, pages 256–263, 2000.

[7] L. Hirschman, A. Yeh, C. Blaschke, and A. Valencia. Overview of biocreative: cretical assessment of information extraction for biology. *BMC Bioinformatics*, 6(Suppl 1)(S1):795–825, May 2005.

[8] S. Kiritchenko, S. Matwin, and A. Famili. Functional annotation of genes using hierarchical text categorization. In *Proceedings of BioLINK SIG: Linking Literature, Information and Knowledge for Biology*, 2005.

[9] M. Light, X. Y. Qiu, and P. Srinivasan. The language of bioscience: Facts, speculations and statements in between. In *Proceedings of BioLink 2004 Workshop on Linking Biological Literature, Ontologies and Databases*, 2004.

[10] A. Moschitti and R. Basili. Complex linguistic features for text classification: A comprehensive study. *Proceedings of the 26th European Conference on Information Retrieval (ECIR)*, pages 181–196, 2004.

[11] S. Ray and M. Craven. Learning statistical models for annotating proteins with function information using biomedical text. *BMC Bioinformatics*, 6(Suppl 1)(S18):291–301, May 2005.

[12] S. B. Rice, G. Nenadic, and B. J. Stapley. Mining protein function from text using term-based support vector machines. *BMC Bioinformatics*, 6(Suppl 1)(S22):291–301, May 2005.

[13] M. Ruiz and P. Srinivasan. Hybrid hierarchical classifiers for categorization of medical documents. *Proceedings of the American Society for Information Science and Technology*, 2003.

[14] M. E. Ruiz and P. Srinivasan. Hierarchical text categorization using neural networks. *Information Retrieval*, 5(1):87–118, 2002.

[15] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[16] A. K. Sehgal and P. Srinivasan. Retrieval with gene queries. *BMC Bioinformatics*, 7(220), April 2006.

[17] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. *Proceedings of the 1996 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, 1996.

[18] W. Wibowo and H. Williams. Minimising errors in hierarchical web categorisation. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM) 2002*, pages 525–531, 2002.

[19] H. Xie, A. Wasserman, Z. Levine, A. Novik, V. Grebinshy, A. Shoshan, and L. Mintz. Large scale protein annotation through gene ontology. *Genome Research*, 12:785–794, 2002.