BIOINFORMATICS ORIGINAL PAPER
Vol. 24 no. 21 2008, pages 2512–2517
doi:10.1093/bioinformatics/btn463

Data and text mining

Improving subcellular localization prediction using text classification and the gene ontology

Alona Fyshe, Yifeng Liu, Duane Szafron*, Russ Greiner and Paul Lu

Department of Computing Science, University of Alberta, Edmonton, AB, T6G 2E8, Canada

Received on July 23, 2008; revised on August 21, 2008; accepted on August 25, 2008

Advance Access publication August 26, 2008

Associate Editor: Martin Bishop

ABSTRACT

Motivation: Each protein performs its functions within some specific locations in a cell. This subcellular location is important for understanding protein function and for facilitating its purification. There are now many computational techniques for predicting location based on sequence analysis and database information from homologs. A few recent techniques use text from biological abstracts: our goal is to improve the prediction accuracy of such text-based techniques. We identify three techniques for improving text-based prediction: a rule for ambiguous abstract removal, a mechanism for using synonyms from the Gene Ontology (GO) and a mechanism for using the GO hierarchy to generalize terms. We show that these three techniques can significantly improve the accuracy of protein subcellular location predictors that use text extracted from PubMed abstracts whose references are recorded in Swiss-Prot.

Contact: duane@cs.ualberta.ca

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 INTRODUCTION

There is a flood of biological data from laboratories around the world. To deal with this overload, there are now many computational methods that make it easier for fellow researchers to find relevant research results quickly and easily. We have developed a technique that uses the text of journal abstracts available through *Swiss-Prot* (Swiss-Prot, 2008) and *PubMed* (PubMed, 2008) in conjunction with the *Gene Ontology* (*GO*) (Gene Ontology Consortium, 2000) hierarchy to significantly improve text classification for biological journal abstracts.

In general, a text classifier maps each text document into one or more predefined labels. For example, text classification of a newspaper article attempts to predict the appropriate newspaper section label, based on the content of that article—whether it belongs in the sports or world news section. Our goal is to improve text classification in a predictor that maps each relevant *PubMed* abstract into a subcellular location (e.g. nucleus or mitochondrion), so that we can predict the location of a protein based on abstracts written about it, as recorded in a field within the protein's *Swiss-Prot* entry.

Classification of biological abstracts is an interesting specialization of general text classification, since these abstracts contain scientific terminology that is often not understandable by non-scientists. They contain specialized terms and acronyms and terms that have many synonyms. For example, the 'PAM complex', which exists in the mitochondrion of a biological cell, is also referred to by the phrases 'presequence translocase-associated import motor' and 'mitochondrial import motor'. This example also illustrates the fact that biological terms often span word boundaries, which means that their phrasal meaning would be lost if we tokenized the text into individual words.

Text classification has been used in the biological domain before, including its use for subcellular localization prediction. However, no researchers have used the hierarchical structure of the *GO* to improve text classification in the biological domain.

We want to *learn* these text classifiers. Many different learning algorithms have been explored for general text classification (Dumais *et al*., 1998). *Support vector machines* (*SVM*s) (Vapnik, 1995) were found to have the highest precision/recall break-even point (BEP, the point where precision equals recall). Joachims (1998) performed a very thorough evaluation of the suitability of *SVM*s for text, discovering that *SVM*s are perfect for textual data because textual data produces sparse training instances in very high dimensional space.

Soon after Joachims' survey, researchers started using *SVM*s to classify biological journal abstracts. Stapley *et al*. (2002) used *SVM*s to improve the prediction of the subcellular localization of yeast proteins. They created a dataset by mining Medline for abstracts that contained a yeast gene name. Their text classifiers achieved F-measures [see Equation (1)] in the range [0.31–0.80], where p is precision, r is recall, TP is true positives, FP is false positives, FN is false negatives and PNP is number of prediction on positive proteins.

$$F-\text{measure}=\frac{2\times r\times p}{r+p},\quad p=\frac{\text{TP}}{\text{TP}+\text{FP}},\quad r=\frac{\text{TP}}{\text{TP}+\text{FN}+\text{PNP}} \tag{1}$$

They also built a subcellular predictor based on amino acid content that had an F-measure in the range [0.03–0.61]. Using text and amino acid composition together yielded an F-measure in the range [0.33–0.82] with improvements of up to 0.05 over text alone. These results are modest, but prior to Stapley *et al*. (2002), most localization classification systems were built using text rules or were sequence based. This was one of the first applications of *SVM*s to biological journal abstracts and it showed that text and amino acid composition together yield better results than either alone.

In other research, text classification was used to augment subcellular localization predictors for *animal* and *plant* datasets

*To whom correspondence should be addressed.

(Höglund *et al*., 2006). These authors' text classifiers were based on the most distinguishing terms of documents, and they described the output of four protein sequence classifiers on their training data. They measured the performance of their classifier using recall (denoted sensitivity) and precision (denoted specificity). Their text-only classifier for the *MultiLoc animal* dataset had recall in the range [0.51–0.93] and precision in the range [0.32–0.91]. The *MultiLocText* classifiers, which include sequence-based features, have recall [0.82–0.93] and precision [0.55–0.95]. Their overall and average accuracy increased by 16.2 and 9.0% to 86.4 and 94.5%, respectively, on the *PLOC animal* dataset when text was used to augment sequence-based features. Our technique (Fyshe and Szafron, 2006) is motivated by the improvements that Stapley *et al*. (2002) and Höglund *et al*. (2006) saw when they added text classification to other biological features. Our method uses only features extracted from the abstracts, augmented by features from the *GO* to improve text classification; it uses no information from the amino acid sequence. Since the text classifier described in this article has better performance than the text classifier of Höglund *et al*. (2006), and does not depend on other features, it can be used as a 'plug-in' replacement to augment predictors that also use non-textual features such as amino acid composition.

Our technique uses existing sources of well-organized data: *Swiss-Prot*, *PubMed* and the *GO* hierarchy. This research is a novel contribution to the area of text classification for biological journal abstracts as it identifies (i) a mechanism for determining which abstracts should be used (*abstract filtering*) and (ii) two techniques for employing the *GO* as a source of expert knowledge to add features for each protein (*term augmentation*).

In this article, we use the controlled vocabulary of biological terms represented by the *GO* to complement the information present in journal abstracts. Specifically, we improve text classification by effectively using:

- The *GO* as a thesaurus to identify synonyms [*synonym resolution (SR)*].
- The hierarchical structure of the *GO* to generalize specific terms into broad concepts [*term generalization (TG)*].

We show that each of these techniques, *abstract filtering* and *term augmentation* (*SR* or *TG*), *individually* improves subcellular localization predictors.

Although, classification of biological journal abstracts is a challenging problem, a solution would yield important benefits. With sufficiently accurate text classifiers, the journal abstracts referenced from a protein database entry could be used to automatically annotate that protein. Here, we use these techniques to help a text classifier predict the subcellular localization of a protein. However, these ideas could also be used to help other text classifiers predict other annotations (e.g. general function, pathway participation, relation to disease). Biologists could use these prediction annotations to more efficiently identify proteins of interest. In addition, when biologists submit new research articles to databases like *Swiss-Prot*, text classifiers could be used to suggest annotations for approval by the submitting biologist. By spending less time sifting through unannotated proteins, researchers will be able to spend more time performing important experiments and uncovering novel knowledge.

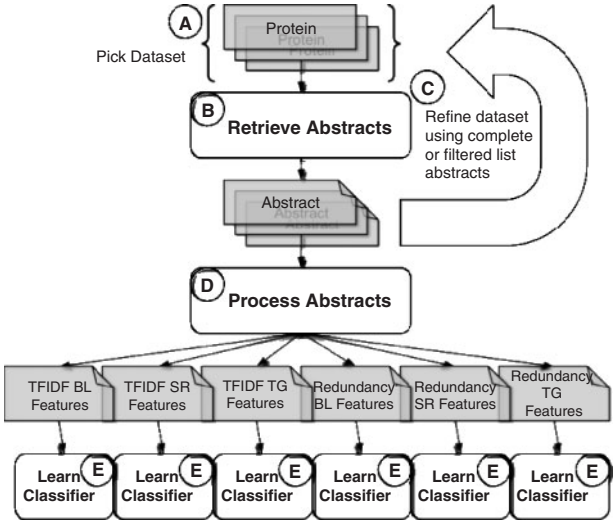

Fig. 1. The workflow for creating the text-based classifiers.

The first step in evaluating our abstract selection mechanism and the *GO* as a knowledge source for enhancing automatic subcellular localization annotation was to create protein datasets. We selected seven initial datasets (step A in Fig. 1). First, we included the *MultiLoc animal* and *plant* datasets (Höglund *et al*., 2006) to allow us to directly compare our results to the best natural language processing (NLP)-only subcellular location predictor we identified in the literature. Second, we used the latest versions of the five larger *Proteome Analyst* (*PA*) datasets (Lu *et al*., 2004) extracted from *Swiss-Prot*.

2 METHODS

Figure 1 presents an overview of the workflow for the experiments outlined in this article.

2.1 The datasets

(Höglund *et al*., 2006) created the *MultiLoc* dataset by searching for phrases in the *Subcellular localization* and *Feature* fields of *Swiss-Prot* version 42.0. The redundancy of the resulting dataset of 9761 proteins was reduced by removing sequences until no pair of sequences shared >80% similarity. Separate *MultiLoc* datasets were constructed from the remaining 5959 proteins. The *plant* dataset contains all proteins (not just plant proteins) that have subcell labels that occur in plants: chloroplast (*ch*), cytoplasm (*cy*), endoplasmic reticulum (*er*), extracellular space (*ex*), Golgi apparatus (*go*), mitochondrion (*mi*), nucleus (*nu*), peroxisome (*pe*), plasma membrane (*pm*) and vacuole (*va*). The *animal* classifier removes *ch* and replaces *va* with lysosomes (*ly*). Therefore, there is significant overlap between the proteins in the *animal* and *plant* datasets. For example, the protein dataset (step A in Fig. 1) has 5447 proteins for the *MultiLoc animal* dataset and 5856 proteins for the *MultiLoc plant* dataset from the total dataset of 5959 proteins.

Lu *et al*. (2004) created the *PA datasets* by collecting a group of proteins whose *Swiss-Prot* entries included a subcellular localization. *PA* parses the *Swiss-Prot* subcellular field to extract a short phrase that identifies the specific subcellular localization (e.g. inner membrane). Subcellular localization categories are organism specific. For example, a bacterial cell does not have a nucleus, and an animal cell does not have a periplasmic space. Therefore, *PA* creates five different datasets, one for each different category of biological cell: *animal*, *green plant*, *fungi*, *gram-negative bacteria* and *gram-positive bacteria*. In this article, we present results for the *animal*

dataset updated using *Swiss-Prot* version 51.3. The results for other *PA* datasets are available online (http://www.cs.ualberta.ca/∼bioinfo/nlp). *PA*'s datasets have a separate 'binary' training file for each label so that a protein can appear with a positive label in more than one file. The *PA* dataset contains 39 385 labeled proteins, including 3384 proteins that localize to more than one location. Unlike the *MultiLoc* datasets, these proteins were divided into five mutually exclusive datasets, based on the protein's organism. For example, the initial dataset shown in Figure 1 has 16 615 proteins for the *PA animal* dataset (including 2337 multilabeled proteins). We created a separate binary classifier for each label of each of the five organism categories, so there were actually nine binary classifiers used for *animal* classification, one for each location (the same locations as the *MultiLoc animal* dataset).

2.2 Retrieving and filtering abstracts

We obtained a protein's abstracts (step B in Fig. 1) using the *PubMed* identifiers recorded in its *Swiss-Prot* database entry. For the *MultiLoc* dataset we used *Swiss-Prot* version 42.0, the same version used by Höglund *et al*. (2006), so that our results could be directly compared to theirs. For the *PA* datasets we used *Swiss-Prot* version 51.3, the newest when the experiments were conducted. We did not include full article text, since it can be difficult to obtain automatically and since previous research has shown that using full text rather than only abstracts does not significantly improve the performance of text classification (Sinclair and Webber, 2004). A protein may have zero or more abstracts and many proteins in *Swiss-Prot* can refer to the same abstract. All proteins with no abstracts were removed from the datasets (step C in Fig. 1). For example, of the 5447 proteins in the *MultiLoc animal* dataset, 5137 have at least one abstract, so we eliminated the other 310 from the dataset.[1] Of the 16 615 proteins in the *PA animal* dataset, 12 261 have at least one abstract so the 4354 with no abstracts were eliminated.

Since different abstracts may have different utilities in predicting subcellular localization, we experimented with *abstract filtering* mechanisms that remove abstracts that might not contribute to good label prediction. For example, an abstract that describes the sequencing of an entire organism is probably not useful, since it is referenced by proteins with different subcellular localizations. An *ambiguous abstract* is one whose removal from a training set results in a predictor whose F-measure is not lower than the F-measure of a predictor that uses that abstract. We investigated techniques for *ambiguous abstract removal*.

The *complete set* is the set of all abstracts referred to by any protein in the protein dataset. The *exclusion set* is the set of abstracts that are each referenced by any group of proteins that have no subcellular labels in common. For example, suppose proteins p_1 and p_2 each reference the same abstract, a_{12}, and p_1 is labeled $\{cy, ex\}$ and p_2 is labeled $\{nu\}$. Then a_{12} would be included in the *exclusion set*, since a_{12} cannot help to differentiate between the labels. However, if instead, p_2 was labeled $\{nu, ex\}$, then a_{12} would not be included in the *exclusion set*, since a_{12} could be describing aspects of proteins related to the common label, *ex*. We compared the approaches of using only the *filtered set* = *complete set—exclusion set* to using the *complete set*. For example, the *MultiLoc* animal dataset has 211 *exclusion set* abstracts and 12 549 *filtered set* abstracts, while the *PA* animal dataset has 122 *exclusion set* abstracts and 25 547 *filtered set* abstracts.

As described later, each predictor uses as features for a given protein, tokens that are extracted from the abstracts that it references. In the *filtered set* domain, any protein that has only *exclusion set* abstracts will have no remaining abstracts and therefore no features. For example, the *MultiLoc animal* dataset has 287 proteins that have only *exclusion set* abstracts and the *PA animal* dataset has 244 proteins with only *exclusion set* abstracts. There are two prediction choices available when a protein has no features. One is to make the prediction based on the prior probabilities—predict the class label with the highest probability. For the *MultiLoc animal* dataset this class is *cy* and for the *PA animal* dataset this class is *extracellular*. The other choice is to make no prediction, which decreases recall, but does not affect precision. We tried both approaches and discovered that F-measure is higher if we make no prediction. We explored whether using a *filtered set* instead of a *complete set* improves accuracy enough on the whole set of proteins to offset the fact that some proteins will have no predictions.

[1]We assume this was done in the Höglund *et al*. (2006) study but were unable to verify it.

2.3 Processing abstracts

To use a classifier, the relevant abstracts for each protein must be assigned a feature vector that can be used by the classifier (step D in Fig. 1). We first create a set of tokens that represent individual words or phrases in an abstract. We use white space or hyphens to determine token boundaries, and then strip all leading and trailing punctuation marks from the tokens. Finally, tokens are stemmed using Porter's stemming algorithm (Porter, 1980) to strip suffixes. In our *baseline* (*BASE*) classifiers, the feature vector has one component (feature) for each unique token that appears in an abstract referenced by any protein in the *filtered set*. For example, there are 46 772 unique tokens for the *PA animal* classifier. We later describe two other classifiers that use additional tokens: SR and *TG*. Our feature vector has one component for each unique token found in any abstract for any protein in the training dataset. The simplest approach is to use a binary feature component for a protein with value 1 if the token appears in one of the abstracts referenced by that protein or 0 if it does not appear. However, it seems more appropriate to use a count of the number of times that the token appears in the selected abstracts as the feature component. In fact, there are more sophisticated NLP techniques called *importance measures* that assign a value to each token that can be used as components in our feature vector.

For this study, we evaluated two importance measures to produce feature vector components, *term frequency–inverse document frequency* (*tfidf*) and *redundancy*, given in Equations (2) and (3), respectively. Note that $f(t_k)$ is the number of times token k appears in an abstract, $f(t_k, d_i)$ is the number of times token k appears in abstract i, N is the total number of abstracts, $d(t_k)$ is the number of abstracts that contain token k and $r(t_k)$, defined in Equation (4), is called empirical entropy.

$$tfidf(t_k) = f(t_k) \times \log \frac{N}{d(t_k)} \tag{2}$$

$$\text{redundancy}(t_k) = f(t_k) \times r(t_k) \tag{3}$$

$$r(t_k) = \log N + \sum_{i=1}^{N} \frac{f(t_k, d_i)}{f(t_k)} \log \frac{f(t_k, d_i)}{f(t_k)} \tag{4}$$

2.4 Synonym resolution

The *GO* hierarchy can be used as a thesaurus for biological words and phrases. For example *GO* encodes the fact that 'metal binding' is a synonym for 'metal ion binding' (Fig. 2). *SR* uses the *GO*'s 'exact synonym' field to find synonyms and incorporates synonym information into the feature vector obtained using the *BASE* processing technique. Specifically, we searched stemmed versions of the abstracts for matches to stemmed *GO* node names or synonyms. If a match was found, the *GO* node name (deemed the canonical representative for its set of synonyms) was added as a token for the protein. However, this node name was prefixed with the string 'go_', so that the *SVM* classifier could differentiate between the case where a *GO* node name appears exactly in text and the case where a *GO* node name is added by *SR*.

For example, in Fig. 3, the phrase 'metal binding' appears in the text. The *GO* hierarchy indicates that this is a synonym of the *GO* node 'metal ion binding'. Therefore the term 'metal ion binding' is added as a 'go' token. This approach combines the weight of several synonyms into one representative, allowing the classifier to more accurately model the author's intent, and to identify multiword phrases that would be otherwise lost during tokenization. Comparing column 5 to column 3 or column 6 to column 4 in Table 1 shows an increase of 3.5% in the number of feature components and the average number of feature vector components per positive training instance for the classifiers constructed from the *PA animal* dataset. We will show that

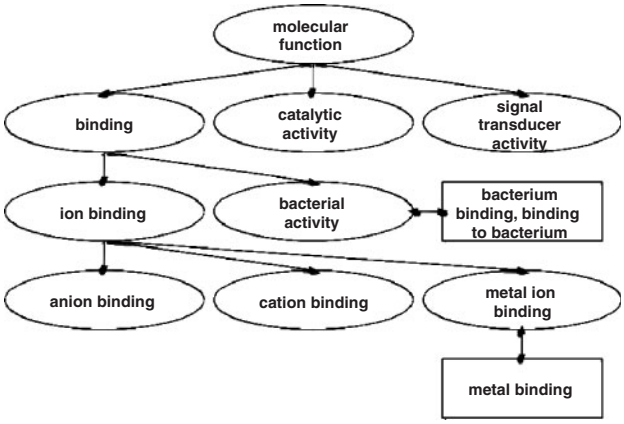

Fig. 2. A subgraph of the *GO* hierarchy. *GO* nodes are shown as ovals, and synonyms are shown as rectangles.

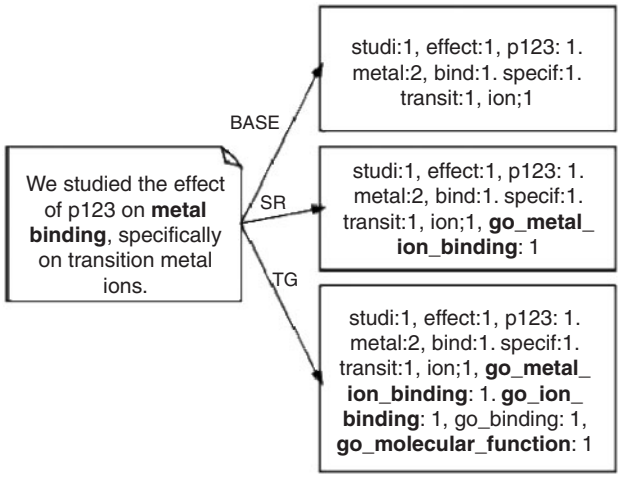

Fig. 3. A sentence that illustrates abstract processing (step D in Fig. 1). Text is white-space tokenized and stemmed. Stop words are removed to create token set used in the *BASE* classifier. *SR* adds a canonical representative for each synonym group. *TG* adds *GO* hierarcy node names that are parents of nodes identified using *SR*. The term 'metal ion binding' is the canonical synonym of 'metal binding' so *SR* adds the *GO* term 'metal ion binding' as a token. *TG* also adds the names of its parent nodes as tokens.

in some cases, this increase in information translates to improved classifier accuracy.

2.5 Term generalization

In order to express the relationships between terms, the *GO* hierarchy is organized in a directed acyclic graph. For example, 'cation binding' is a type of 'metal binding', which is a specific type of 'binding'. This 'is a' relationship is expressed as a series of parent–child relationships (Fig. 2). In *TG*, we first use the *GO* for SR, but we also use the *GO*'s hierarchical structure to generalize specific terms into broader concepts. For *TG*, if a *GO* node name (or synonym) is found in an abstract, then in addition to adding a feature for the canonical synonym, we add a feature for the name of each ancestor (*TG* in Fig. 3). Just as in *SR*, the additional features are prefixed with the string 'go_'. TG further increases the number of feature components and the average number of feature vector components per training instance by another 6.9%, as shown in Table 1.

Table 1. *PA filtered* dataset for the Animal classifiers, where Train Ins is the number of positive training instances, Feature is the number of feature vector components, F/Ins is the average number of feature vector components per positive training instance

Class	Train Ins	*BASE*		*SR*		*TG*	
		Feature	F/Ins	Feature	F/Ins	Feature	F/Ins
cy	2350	21 575	9.18	23 991	10.21	26 559	11.30
er	108	3475	32.18	3798	35.17	4350	40.28
ex	5289	25 355	4.79	27 175	5.14	29 114	5.50
go	28	1686	60.21	1971	70.39	2375	84.82
ly	141	5892	41.79	6395	45.35	7059	50.06
mi	889	10 508	11.82	11 559	13.00	12 918	14.53
nu	3618	23 192	6.41	25 513	7.05	27 708	7.66
pe	108	3603	33.36	3942	36.50	4466	41.35
pm	206	5748	27.90	6375	30.95	7237	35.13
All	12 261	45 197	3.69	48 904	3.99	52 157	4.25

The compartments are cy, er, ex, go, ly, mi, nu, pe and pm. The 12 261 proteins in the dataset include 468 multi-labeled proteins.

TG gives the *SVM* algorithm an opportunity to learn correlations that exist between general terms and subcellular localization even if the general term itself never appears in an abstract as only the names of its more specific children occur. Without TG, the *SVM* has neither concept of the relationship between child and parent terms, nor between sibling terms. For some localization categories more general terms may be the most informative and in other cases specific terms may be best. Because our technique adds features to training instances and never removes any, the *SVM* can assign lower weights to the generalized terms in cases where the localization category is not well characterized by more general terms.

3 RESULTS AND DISCUSSION

Our *BASE*, *SR* and *TG* classifiers were trained and evaluated using cross validation on seven separate datasets: *MultiLoc animal*, *MultiLoc plant*, *PA animal*, *PA plant*, *PA fungi*, *PA gram negative bacteria* and *PA gram-positive bacteria*. For brevity, this article contains F-measure scores for the *MultiLoc animal* and *PA animal* datasets using *tfidf*. Results using the other datasets (including precision and recall scores) and using *redundancy* as well as *tfidf* are available online (http://www.cs.ualberta.ca/~bioinfo/nlp). For the *MultiLoc* datasets, we used 5-fold cross validation so the results could be compared directly to Höglund *et al*. (2006). For the *PA* datasets we used 10-fold cross validation.

Table 2 shows the F-measure scores for the *MultiLoc animal* dataset and *Swiss-Prot* version 42.0, including the text-only classifier (denoted *TEXT*) reported by Höglund *et al*. (2006). The best score is bolded in each row. The *BASE-complete* predictor is comparable to the *TEXT* classifier. The F-measures differ in the range -15.1% (*mi*) to +20.6% (*ly*) with an overall difference of +0.05%. This overall score is not the simple average of the individual scores. Instead, it is the mean of the five aggregate F-measures, each taken over the single confusion matrix constructed for all predictions. For the *TEXT* predictor, we did not have fold data so the overall score is the aggregate F-measure over the single confusion matrix for all predictions.

We used a one-sample t-test to compare the mean F-measure of the five overall *BASE-complete* scores (df = 4) to the overall *TEXT* score. The t-score was 0.3465 ($p = 0.3732$), indicating that we only have 62.68% confidence that the F-measure of the *BASE-complete* classifier is higher than the F-measure of the *TEXT* classifier, so we have established that it is comparable—not better. This indicates that our *BASE* predictor is not a 'straw man', since it is as good as the best-known predictors appearing in the literature that are based on abstracts alone. The key differences between *BASE-complete* and *TEXT* are:

(1) After white space tokenization and stemming, *TEXT* uses only a subset of machine-learned distinguishing terms (∼800 terms for 10 000 abstracts). *BASE* uses all terms (stemmed).

(2) *TEXT* uses a probabilistic term weighting scheme, where the weight of a distinguishing term is assigned the ratio of its occurrence count in the abstract to the sum of the occurrence counts of all distinguished terms in abstracts referenced by the protein. We use standard NLP algorithms: *tfidf* and *redundancy*.

(3) *TEXT* uses a single multilabel classifier that predicts a single label. We use a set of binary classifiers, one for each label in the ontology of the category (*animal* or *plant*). This allows us to make correct predictions for proteins that localize to more than one location.

Our goal is to evaluate the use of abstract *filtered sets* as an ambiguous abstract removal technique and the use of *GO* terms to add effective features. First, we show that ambiguous abstract removal using a *filtered set* almost always produces a predictor with higher F-measure than a *complete set* predictor that removes no abstracts. However, we discuss a specific situation where this assertion fails, meaning that it is better to use the *complete set* of abstracts rather than the *filtered set*. Table 2 shows that a *filtered set* improves the F-measure score of all *BASE*, *SR* and *TG* predictors except for the *nu* class. The overall improvements are 6.9, 6.4 and 5.3%, respectively. We used a one sample t-test to test whether the overall F-measure for the *filtered set* are at least 5% higher than the overall F-measure for the *complete set* across the 5-folds and three types of classifiers (*BASE*, *SR* and *TG*—fifteen data points in total) by comparing '*filtered set* score minus *complete set* score minus 0.05' to 0. This test establishes, with 99.8% confidence ($p = 0.001989$, df = 14, $t = 3.44$) that the *filtered set* F-measure is at least 5% better than the *complete set* F-measure. A test for predicting when to use a *complete set* for a particular class instead of using the *filtered set* is given later in this article.

Table 2 also shows that using the *GO* for *SR* or SR plus *TG* improves predictions over *BASE* when the abstract set is fixed in almost all cases. Out of 18 cases (nine complete and nine filtered), only two resulted in lower F-measures: *SR-complete* for *er* (0.570–0.554) and *TG-filtered* for *mi* (0.746–0.727). In all other cases, using the *GO* improves performance. The overall F-measure increases by 1.6% (0.731–0.747) for *SR-complete*, 2.6% (0.731–0.757) for *TG-complete*, 1.1% (0.800–0.811) for *SR-filtered* and 1.0% (0.800–0.810) for *TG-filtered*. We used a one sample t-test to test whether the overall F-measure for the *SR* and *TG* predictors are at least 1% higher than the overall F-measure for the *BASE* predictor across the 5-folds and two dataset abstractions (*complete set* and *filtered set*—10 data points each for *SR* and *TG*) by comparing '*SR or TG* score—*BASE* score—0.01' to 0. For *SR*, we measured 92% confidence ($p = 0.08207$, df = 9, $t = 1.5148$) that the *SR* F-measure is at least 1% better than the *BASE* F-measure. For *TG*, we measured 95% confidence ($p = 0.04599$, df = 9, $t = 1.8855$) that the *TG* F-measure is at least 1% better than the *BASE* F-measure. The difference between overall F-measure scores for the *SR* and *TG* predictors was not statistically significant.

Table 2. F-measure scores for the *MultiLoc animal* dataset

Class Abstract	*TEXT* Comp	*BASE* Comp	*BASE* Filtered	*SR* Comp	*SR* Filtered	*TG* Comp	*TG* Filtered
cy	0.610	0.700	0.739	0.712	**0.751**	0.715	0.746
er	0.580	0.570	0.649	0.554	0.662	0.576	**0.663**
ex	0.770	0.820	0.833	0.836	0.845	0.846	**0.851**
go	0.550	0.702	0.724	0.713	0.755	0.740	**0.766**
ly	0.450	0.656	0.664	0.682	0.669	0.662	**0.687**
mi	**0.790**	0.639	0.746	0.654	0.750	0.648	0.726
nu	0.770	0.728	0.718	0.750	0.731	**0.780**	0.727
pe	0.730	0.695	0.754	0.725	**0.773**	0.695	0.760
pm	0.850	0.779	0.852	0.800	0.860	0.813	**0.871**
Overall	0.726	0.731	0.800	0.747	**0.811**	0.757	0.810

TEXT is the classifier described in Höglund *et al*. (2006). The abstracts line indicates whether a *filtered set* of abstracts was used (*filtered*) or a *complete set* of abstracts (*comp*). *BASE*, *SR* and *TG* used *tfidf*. The compartments are cy, er, ex, go, ly, mi, nu, pe and pm. The bolded value in each line is the highest F-measure for that cellular compartment.

Combining the benefits of *filtered sets* and the *GO* results in an overall 8.0% improvement in F-measure, when *SR* is used and 7.9% when *TG* is used. A two-sample t-test shows that *SR-filtered* has 4% higher F-measure than *BASE-complete* with confidence 96% ($p = 0.03913$, df = 7.98, $t = 2.019$). *TG-filtered* has 4% higher F-measure than *BASE-complete* with confidence 96% ($p = 0.04091$, df = 7.98, $t = 1.9903$). These results indicate that, in general, if a predictor uses NLP techniques on biological abstracts then *filtered sets*, and using the *GO* are effective in improving F-measure.

The results of similar experiments with the *PA dataset* using *Swiss-Prot* version 51.3 and the *tfidf* importance measure are shown in Table 3 (without an entry for *TEXT*). However, there are now 10-folds so the number of data points is doubled. Using the *filtered set* of abstracts rather than the *complete set* results in a 4% higher overall F-measure score (4.7% for *BASE*, 5.0% for *SR* and 4.8% for TG) with confidence 99% ($p = 0.0026$, df = 29, $t = 3.0274$). Of the 27 comparisons, only one has a *complete set* F-measure higher than *filtered set* F-measures (*SR* for *ly*) it is not statistically significant.

For *SR*, we measured 99.98% confidence ($p = 0.0001599$, df = 19, $t = 4.383$) that the *SR* F-measure is better than the *BASE* F-measure (0.4% for a *complete set* predictor and 0.7% for a *filtered set* predictor). For *TG*, we measured 99% confidence ($p = 0.0009$, df = 19, $t = 3.6344$) that the *TG* F-measure is better than the *BASE* F-measure (0.6% for a *complete set* predictor and 0.7% for a *filtered set* predictor). Combining the benefits of *filtered sets* and the *GO* results in an overall 5.4% improvement in F-measure, when either *TG* or *SR* is used. A two sample t-test shows that *SR-filtered* has 3% higher F-measure than *BASE-complete* with confidence 99% ($p = 0.01498$, df = 17.79, $t = 2.3592$). *TG-filtered* has 3% higher F-measure than *BASE-complete* with confidence 99% ($p = 0.01370$,

Table 3. *F*-measure scores for the *PA animal* dataset

Class Abstract	*BASE* Comp	*BASE* Filtered	*SR* Comp	*SR* Filtered	*TG* Comp	*TG* Filtered	*IMPR* TG-f
cy	0.706	0.746	0.715	0.759	0.716	**0.760**	5.4%
er	0.595	0.699	0.632	0.705	0.712	**0.735**	14.0%
ex	0.927	0.953	0.927	**0.956**	0.928	**0.956**	2.9%
go	0.290	0.376	0.370	0.422	0.316	**0.426**	13.6%
ly	0.789	0.797	0.801	0.800	0.807	**0.809**	2.0%
mi	0.790	0.822	0.777	**0.831**	0.776	0.828	3.8%
nu	0.847	0.893	0.851	**0.901**	0.852	**0.901**	5.4%
pe	0.761	0.793	0.800	0.832	0.843	**0.851**	9.0%
pm	0.680	0.717	0.708	**0.739**	0.710	0.734	5.4%
Overall	0.843	0.890	0.847	**0.897**	0.849	**0.897**	5.4%

The complete (Comp) and filtered (Filtered) abstract sets for the BASE, SR, and TG predictors. The Improve column (IMPR) indicates the amount by which *TG filtered* improved over *BASE complete.* The compartments are cy, er, ex, go, ly, mi, nu, pe and pm. The bolded value in each line is the highest F-measure for that cellular compartment.

Table 4. Numbers of proteins with only *exclusion set* abstracts, number of abstracts in the *exclusion set* and the ratio of these two numbers in the *MultiLoc animal* dataset and the *PA animal* dataset

Class	*MultiLoc animal*			*PA animal*		
	Proteins	*Abstracts*	*Ratio*	*Protein*	*Abstracts*	*Ratio*
cy	33	156	0.21	77	46	1.67
er	22	50	0.44	2	4	0.50
ex	0	46	0.00	41	25	1.64
go	9	33	0.27	0	0	–
ly	0	17	0.00	2	3	0.67
mi	56	94	0.60	66	49	1.35
nu	116	76	1.53	57	50	1.14
pe	10	29	0.34	3	5	0.60
pm	41	102	0.40	3	4	0.75
Overall	287	211	1.36	244	122	2.00

The compartments are cy, er, ex, go, ly, mi, nu, pe and pm.

df = 17.79, $t = 2.405$). These results confirm that *filtered sets*, and using the *GO* provide a significant improvement when using NLP techniques based on biological abstracts.

To discover the reason why the *complete set* of abstracts was better than the *filtered set* for the *nu* class in the *MultiLoc animal* dataset, we examined many statistics based on the number of abstracts in the *exclusion set* for each class and the number of proteins with labels in each class that yield no features when the abstract *filtered set* is used instead of the abstract *complete set*. Table 4 shows that *nu* (*Multiloc animal*) is the only class in which the ratio of proteins removed to abstracts removed is larger than the overall ratio for the *filtered set*. Therefore, we conjecture that a class ratio that is higher than the overall ratio should be used as the criteria for determining whether to make an exception to using the *filtered set* instead of the *complete set*. We then computed the ratios for the *PA animal* dataset that are shown in Table 4. The ratios suggest that no *PA animal complete set* classifier should have a (statistically significant) higher *F*-measure than the *PA animal filtered set* classifier for the same class. Table 3 provides evidence for our conjecture.

The NLP-based classifiers described in this article can be used to augment other non-NLP predictors. We are currently using our new NLP techniques to improve the coverage and accuracy of the PA family of predictors (Lu *et al*., 2005; Szafron *et al*., 2004). There are several ways that classifiers can be combined into an ensemble predictor and it is not clear yet which technique will provide the best overall win in terms of increased coverage and accuracy.

In summary, using the *GO* for SR and TG is a useful mechanism for improving text classification using biological abstracts. In addition, we recommend *ambiguous abstract removal* using a *filtered set* for those predictors whose ratio of empty feature vectors per abstract removed is lower than the overall ratio for all of the predictors.

ACKNOWLEDGEMENTS

We would like to thank Greg Kondrak, Colin Cherry, Shane Bergsma and the whole NLP group at the University of Alberta for their suggestions and guidance, as well as Kurt McMillan, Jordan Patterson, Pandora Lam, Cam MacDonell and the rest of the Proteome Analyst team.

Funding: Natural Sciences and Engineering Research Council of Canada (NSERC); Informatics Circle of Research Excellence (iCORE); Alberta Ingenuity Centre for Machine Learning (AICML).

Conflict of Interest: none declared.

REFERENCES

Dumais,S.T. *et al*. (1998) Inductive learning algorithms and representations for text. In *Proceedings of 7th International Conference on Information and Knowledge Management CIKM*, pp. 148–155.

Fyshe,A. and Szafron,D. (2006) Term generalization and synonym resolution for biological abstracts: using the gene ontology for subcellular localization prediction. In *BioNLP Workshop at HLT-NAACL*, pp. 17–24.

Gene Ontology Consortium. (2000) Gene ontology: tool for the unification of biology. *Nat. Genet.*, **25**, 25–29.

Höglund,A. *et al*. (2006) Significantly improved prediction of subcellular localization by integrating text and protein sequence data. In *Pac. Symp. Biocomput.*, 16–27.

Joachims,T. (1998) Text categorization with support vector machines: learning with many relevant features. In *ECML*, 137–142.

Lu,P. *et al*. (2005) PA-GOSUB: a searchable database of model organism protein sequences with their predicted go molecular function and subcellular localization. *Nucleic Acids Res.*, **33**, D147–D152.

Lu,Z. *et al*. (2004) Predicting subcellular localization of proteins using machine-learned classifiers. *Bioinformatics*, **20**, 547–556.

Porter,M.F. (1980) An algorithm for suffix stripping. *Program*, **14**, 130–137.

PubMed. (2008) Available at http://www.ncbi.nlm.nih.gov/sites/entrez (last accessed date 2 September, 2008).

Sinclair,G. and Webber,B. (2004) Classification from full text: a comparison of canonical sections of scientific papers. *COLING (NLPBA/BioNLP)*, 69–72.

Stapley,B.J. *et al*. (2002) Predicting the sub-cellular location of proteins from text using support vector machines. In *Pac. Symp. Biocomput.*, 374–385.

Szafron,D. *et al.* (2004) Proteome analyst: custom predictions with explanations in a web-based tool for high-throughput proteome annotations. *Nucleic Acids Res.*, **32**, W365–W371.

Swiss-Prot. (2008) *Swiss-Prot Protein Knowledgebase*. Swiss Institute of Bioinformatics.Available at http://ca.expasy.org/sprot/ (last accessed date 2 September, 2008).

Vapnik,V.N. (1995) *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, USA.