

Learning to Find Relevant Biological Articles Without Negative Training Examples

Keith Noto, Milton H. Saier Jr., and Charles Elkan

University of California, La Jolla CA 92093

`knoto@cs.ucsd.edu,msaier@biomail.ucsd.edu,elkan@cs.ucsd.edu`

Abstract. Classifiers are traditionally learned using sets of positive and negative training examples. However, often a classifier is required, but for training only an incomplete set of positive examples and a set of unlabeled examples are available. This is the situation, for example, with the Transport Classification Database (TCDB, www.tcdb.org), a repository of information about proteins involved in transmembrane transport. This paper presents and evaluates a method for learning to rank the likely relevance to TCDB of newly published scientific articles, using the articles currently referenced in TCDB as positive training examples. The new method has succeeded in identifying 964 new articles relevant to TCDB in fewer than six months, which is a major practical success. From a general data mining perspective, the contributions of this paper are (i) devising and evaluating two novel approaches that solve the positive-only problem effectively, (ii) applying support vector machines in a state-of-the-art way for recognizing and ranking relevance, and (iii) deploying a system to update a widely-used, real-world biomedical database. Supplementary information including all data sets are publicly available at www.cs.ucsd.edu/users/knoto/pub/ajcai08.

1 Introduction

The transport classification database, or TCDB (www.tcdb.org), is an online database which contains sequence, structural, and functional information about proteins that relate to transport across cell membranes in a variety of organisms, categorized into over 550 families of proteins [11]. TCDB is widely used, averaging over 50 different users per day from research institutions all over the world. TCDB defines and implements the transport classification system [10] for categorizing transport proteins, which was adopted by the International Union of Biochemistry and Molecular Biology as the international standard in 2002.

As of October 15, 2007, the start of the project described in this paper, the data contained in TCDB were compiled from 3,403 publications in over 200 different journals. Our goal is to help keep TCDB updated with all relevant new information related to transport proteins. The sources of information that we consider in this paper are recent articles published in biological journals. Information that goes into TCDB is verified by a human expert before inclusion. Accordingly, our goal is not full automation. However, even if we restrict the set

of journals to those already referenced in TCDB, there are several thousand articles published each month—too many for a human expert to examine. Of these, only a small percentage are relevant to TCDB. Our goal is to identify these relevant articles automatically, as soon as possible after their abstracts are published in Medline.

The procedure for providing the expert with candidate articles is to (i) learn a model for ranking future articles, (ii) use the model to rank articles from a candidate set, and (iii) provide the expert with as many articles as the expert is able to screen (we call this number k —typically a small percentage of the candidate articles). Therefore, the technical task we consider in this paper is to learn a classifier for Medline abstracts that maximizes the number of relevant articles in the set of size k shown to the human expert.

Of course, this task is not unique to TCDB. The *Nucleic Acids Research* journal lists over 1,000 specialized databases as of the January, 2008 update [6]. With over 50,000 citations currently added to Medline each month, the need for methods to screen and select documents automatically is increasing. Similar tasks have been investigated before [4, 16, 17], but previous approaches assume the availability of labeled negative examples as well as of labeled positive examples. Below, we investigate the implications of only having positive and unlabeled training examples available.

We use a support vector machine (SVM) as our classifier. The available data for training consist of (i) articles already referenced in TCDB and (ii) recent articles published in the same journals. In principle, all 18 million Medline documents could be unlabeled training examples, but in practice we use a sample of these that is limited to recent articles from journals known to contain relevant articles. Articles in the first training set are labeled positive, but articles in the second set are unlabeled. Thus, we face the “positive-only” issue: We wish to learn a classifier that discriminates positives from negatives, but we have no specific negative training examples. We do however know that most of the unlabeled examples are negative.

Our human expert is able to screen only a limited number of documents. Given a set of unclassified abstracts (*i.e.* a test set of future articles), our classifier scores them according to their likelihood of being relevant to the TCDB database. We then deliver the highest-scoring ones to the human expert for screening. We wish to minimize the the number of delivered articles that turn out to be irrelevant. Thus the objective function to maximize is not classification accuracy on the whole test set, but rather precision on a small high-scoring subset of the test set.

Formally, let x represent the features of an article, and let $y = 1$ represent the fact that the article is relevant (in this case, to TCDB). We wish to learn a scoring function $f(x)$ that, given a set of examples $T = \{x_1, x_2, \dots, x_n\}$, maximizes

$$precision(T, f, k) = \frac{1}{k} \sum_{i=1}^n I(rank(x_i, f) \leq k \wedge y_i = 1) \quad (1)$$

where the indicator function $I(\cdot)$ returns 1 if its argument is true, 0 otherwise, and $rank$ gives the position of x_i if T is sorted by descending $f(x)$; that is, $rank(\arg \max_x f(x), f) = 1$. In words, Equation (1) gives the proportion of examples in the top k according to f that are positive, which is called the precision at k . This is our objective function and is therefore used throughout this paper to evaluate our models.

The fact that we are given only positive and unlabeled examples at the outset from which to learn f is the main difference between ours and previous approaches which learn from positive and negative training examples [4,16,17]. Our approach and previous approaches to similar tasks, which learn naïve Bayes [17], and SVM models [4,16], as well as nonstandard classifiers [3,4] from positive and negative training examples.

In Section 2, we describe how to train an SVM to achieve good classification accuracy. In Section 3, we explain two solutions to the positive-only issue. The first solution iteratively retrains a classifier, while the second solution relies on a new mathematical result. In Section 4, we describe how we estimate the recall of our models without negative examples. In Section 5, we describe how the final trained classifier has been deployed successfully to find hundreds of new relevant articles for TCDB.

2 Data and Training

The data set that we use to develop and evaluate our methods consists of:

- 3,403 articles that appear in Medline and were referenced in TCDB as of October 15, 2007—these are our positive examples, and
- 16,341 unlabeled articles published recently in those journals referenced in TCDB.

The abstracts of these articles are obtained using the following procedure: First, we download the article data using NCBI’s entrez programming utilities.¹ The positive examples are downloaded individually using their PubMed ID number.² The unlabeled examples are downloaded using the query term “JOURNAL [TA] & MINDATE= ... & MAXDATE= ...” for each journal in TCDB, for the date ranges of October 1 to 31, 2007 (retrieved on December 12, 2007) and November 1 to December 20, 2007 (retrieved on December 20, 2007). This results in a subset of the articles in the PubMed ID range 17902656 to 18092361. The subset contains 16,341 articles because it is restricted to the articles that appear in certain journals and have PubMed dates in the given range associated with them. We represent each article as a vector of features which are to the log-scaled counts of word stems that appear in the article’s abstract. We represent each article as a vector of features corresponding to the words in the article’s abstract. We

¹ See eutils.ncbi.nlm.nih.gov/entrez/query/static/efsearch_help.html.

² All Medline documents have a unique ID number in PubMed, which is NCBI’s public interface to Medline.

apply the Porter stemming algorithm [14] to the abstract text, count the use of each word in each abstract using MALLET [13], and limit the vectors to words that appear at least three times in the corpus. This reduces the vocabulary size in our data set by 47% (17,060 words appear once, 6,534 words appear twice, and we retain a vocabulary of 26,364 words that appear at least three times). Finally, we represent each word by the value $\log(1 + n_i)$, where n_i is the number of times word i appears in an abstract, and we normalize the vector for each article to have unit Euclidean length, *i.e.* to lie on the unit sphere. We do not use information associated with the articles (journal names, author names, *etc.*) other than the abstract text. We verify that the vocabulary does not contain any obvious “leaker” words, *i.e.* terms that discriminate between the training subsets but are not valid predictors for future articles. In particular, the trained models do not use article dates, directly or indirectly, to make predictions. Recall the evaluation metric in Equation (1). For this data set, we choose $k = 3000$, this being about the number of articles the human expert is able to screen in the time available, considering that the positive set is already labeled.

We label training examples that are obtained from TCDB as positive, and we tentatively label the other examples as negative, even though some of them must actually be positive. Based on an informal examination of the articles during the course of this research, we estimate this fraction at about 2.3%.

We use a support vector machine (SVM) as our classification method. Specifically, we use the *svm-light* implementation of soft-margin SVMs [8]. Alternatives to SVMs include maximum entropy and naïve Bayes models. One reason that we choose to use SVMs is because they are a discriminative (as opposed to generative) method. Maximum entropy is also discriminative, but SVMs have two additional advantages. First, their training objective is to maximize the margin in feature space between positive and negative examples, which gives them a small but useful boost in accuracy compared to methods like maximum entropy that maximize variants of log likelihood instead. Second, SVMs facilitate the use of nonlinear kernels. There are methods for learning SVMs that directly optimize nonlinear performance measures like Equation (1) [9], but for simplicity we use standard SVMs. We do not, however, use the SVM’s natural threshold directly. We use the SVM to generate a ranking of examples, and we deliver the top-ranked k examples to the human expert. To evaluate trained classifiers fairly, we use ten-fold cross-validation. For each of the ten test folds, we deliver the highest-scoring 300 articles (for a total of $k = 3000$) to the human expert for inspection.

We train all SVMs using a quadratic kernel. We choose this kernel because it allows interactions between words to influence predictions, in addition to individual words. In preliminary informal experiments, it performs as well as or better than other standard kernels on our task. There is no known principled way to select an appropriate value for the SVM soft-margin penalty parameter C (the tradeoff between margin width and training set accuracy). The optimal value of C for our task may be different from its optimal value in other tasks, because some of the unlabeled examples are relevant to TCDB and will be mis-

classified when tentatively labeled as negative. Therefore, to select C we use nested cross-validation (*i.e.* tuning set cross-validation within testing set cross-validation). We search for the best value of C by starting at $C = 1$, and trying both higher and lower values until the tuning set precision begins to decrease. To evaluate each value of C , we use four-fold cross-validation and consider powers of 10. Figure 1 shows the tuning set precision for each of ten testing folds. Our approach chooses $C = 1$ in nine of ten folds, and $C = 10$ once.

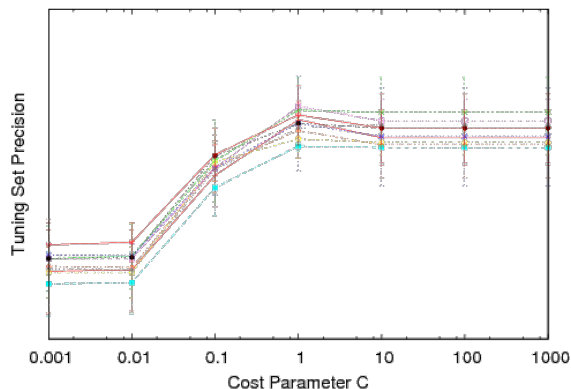


Fig. 1. The tuning set precision associated with the value of the cost parameter in our SVM models. Each line corresponds to one of ten folds.

3 Learning Without Negative Examples

As explained above, a major difficulty of our task is that we do not begin with any set of examples known to be genuinely negative. This difficulty is not unique to our task: In many application domains only positive examples are available naturally for training. Often, negative examples in these domains are ubiquitous, but it is too time-consuming or costly to label and organize them explicitly. The issue of learning a classifier from unlabeled and positive training examples, instead of from negative and positive examples, has been investigated before, *e.g.* [2, 12, 15]. However, the issue has not yet been investigated in the context of the task of identifying articles that are relevant to a biomedical database. We provide a comparison with the aforementioned methods in [5], and we believe that our recent formalization (explained in detail Section 3.2) demonstrates that our approaches are well-suited to this task.

3.1 Iterative relabeling

Our first approach to this issue is to find genuine labels for unlabeled examples predicted to be positive, following an iterative technique introduced by Das *et al.* [1] for the application of identifying relevant database records. That is, we show the unlabeled portion of the top k predictions to a human expert, who assigns each article to one of the following categories:

- Accepted (and added to TCDB)
- Not relevant (the article is not about transport proteins)
- Relevant, but not accepted

The “relevant, but not accepted” category contains articles that are about transport proteins, but that contain information already in TCDB from a different article, or information of low importance and therefore excluded at the discretion of the human expert. Examples in these three categories are labeled positive, labeled negative, and discarded, respectively. After relabeling, we learn a new SVM model and repeat the process.

We perform this procedure until there are no unlabeled predictions in the top k , or until a fixed number of iterations has been reached. The human expert has fewer articles to screen each time, because most of the top k are the same articles each time. Table 1 shows how many examples fall in each category during the relabeling process. Note that fewer and fewer articles that appear in the top k are marked as “not relevant” to TCDB. Very few of the remaining unlabeled articles are likely to be relevant to TCDB. For this reason, we did not ask the human expert to spend the time to do more than two iterations.

Table 1. The number of unlabeled examples that fall into the top k predictions and the assignments given to them by the human expert during each iteration of relabeling.

Iteration	Unlabeled Examples in Top k	Accepted into TCDB	Not Relevant	Relevant, but not Accepted
1	386	182	126	78
2	49	18	16	15

Although the human expert makes a distinction between accepting an article to be added into TCDB and classifying it as relevant, but not accepted, we do not attempt to distinguish between these two article types automatically. The expert makes this distinction because the task is to find new and important information for TCDB. However, articles of both types are indeed about transport proteins. Many relevant but rejected articles are important, and are not accepted only because they contain information that is already present in TCDB.

3.2 Transforming predicted relevance scores

The previous section explains an iterative process that uses a human expert to provide the true labels for unlabeled training examples that are tentatively

predicted to be relevant. This process does improve the accuracy of the final classifier obtained, but only slightly. In this section, we employ a recent mathematical result to explain why the improvement is slight [5].

Let x represent an article in the entire Medline universe, and let s be a random variable such that $s = 1$ means that the article x is selected as a positive training example, *i.e.* x is one of the articles cited currently in TCDB. Let $y = 1$ mean that the article x is truly relevant to TCDB, whether or not x is currently cited in TCDB. Our goal is to find precisely those x for which $y = 1$ but $s \neq 1$.

Assume that every relevant article in the universe has an equal chance of being included in TCDB already. Mathematically, this assumption is that $P(s = 1|y = 1, x)$ is a constant. Of course, newly published articles in the universe cannot have an equal chance of being already in TCDB. However, since our methods do not use dates in any way, this assumption is reasonable when x refers to only the utilized characteristics of an article.

The goal is to learn a function $f(x)$ such that $f(x) = P(y = 1|x)$ as closely as possible. Suppose we provide the labeled data ($s = 1$) and unlabeled data ($s = 0$) as inputs to a standard training algorithm. This algorithm will yield a function $g(x)$ such that $g(x) = P(s = 1|x)$ approximately. The following lemma shows that $f(x)$ is a transformed version of $g(x)$.

Lemma 1: Suppose $P(s = 1|y = 1, x)$ is a constant. Then $P(y = 1|x) = P(s = 1|x)/P(s = 1|y = 1)$.

Proof: The assumption that $P(s = 1|y = 1, x)$ is a constant implies that $P(s = 1|y = 1, x) = P(s = 1|y = 1)$ for all articles x . So

$$\begin{aligned} P(s = 1|x) &= P(y = 1 \wedge s = 1|x) && (\text{because } s = 1 \text{ implies } y = 1) \\ &= P(y = 1|x)P(s = 1|y = 1, x) && (\text{rewritten}) \\ &= P(y = 1|x)P(s = 1|y = 1) && (\text{assumption}) \end{aligned}$$

The result follows by dividing both sides by $P(s = 1|y = 1)$. ■

Although the proof above is simple, this result was published only recently [5]. The reason that the result is novel is perhaps that although the scenario of learning from unlabeled and positive examples has been discussed in many previous papers, it has not previously been formalized using the s random variable.

The lemma says that $P(y = 1|x)$ is a constant times $P(s = 1|x)$. Hence, sorting examples x by their predicted value $P(s = 1|x)$ gives the *same ranking* as sorting by $P(y = 1|x)$. Suppose we use an SVM to sort examples. Although the scores given by such a classifier are not correct probabilities $P(s = 1|x)$, the best estimators of these probabilities are monotonically increasing functions of the SVM scores. Hence, sorting examples by their SVM scores also in principle gives the same ranking as sorting examples by $P(y = 1|x)$.

Given the argument above, the top k abstracts identified by an SVM trained on the unlabeled and positive training examples should be the same as the top k identified by an SVM trained on negative and positive examples. In other words, relabeling should have no effect on the outcome of the model.

Table 2 shows how our classifier improves during the relabeling process. The numbers in this table are computed using knowledge of positive and negative

Table 2. Distribution of the top- k predictions for multiple iterations of relabeling.

Iteration	Relevant	Not Relevant
Before Relabeling	2873	127
After 1 Iteration	2898	102
After 2 Iterations	2896	104

labels acquired via the relabeling process. Comparing lines in the table shows that the relabeling process only leads to a small increase in success (indeed, the results of the second iteration are slightly worse). The lemma above can be viewed as an explanation of why this increase is small. It implies that if they have limited resources, future researchers faced with a database curation task similar to the addressed in this paper will not lose much by not identifying genuine negative training examples manually.

4 Estimating Recall

Although we can estimate the precision of our approach by examining the labels that the human expert assigns to the high-scoring articles predicted by our model, we cannot estimate the *recall* of our models without knowing which of the unlabeled test set examples are false negatives.

To estimate the recall of our models, we take advantage of a separate set of relevant articles that are chosen by the human expert directly from the literature. We then assume that all of the articles from those same journal issues which are *not* selected by the human expert are genuine negative examples. The set of positive examples consists of 37 articles from a total of 44 issues of three particular journals, *the Journal of Bacteriology*, *the Journal of Biological Chemistry*, and *Nature*, between the dates of September 7 and December 14, 2007. The set of negative examples consists of 370 articles from the same journal issues. We train our model on a set of positive articles that are referenced by TCDB, and a set of unlabeled articles from the same journals referenced by TCDB, all of which have a PubMed date strictly prior to September 7, 2007.

Results are shown in Figure 2. It is difficult to decide an appropriate value for the threshold k to use in this figure. We choose $k = 39$ based on the following reasoning. The 418 articles from three journals span approximately a period of time when we delivered 600 articles to our human expert from a total of 102 journals (the three mentioned above, and 99 more). So, we use our learned model to rank all 15,125 articles from these 102 journals in this time period. We find that of the top 600 predictions across all journals, $k = 39$ concern papers in the three journals of interest. In detail, the 600 predictions break down as follows:

	Positive examples from three journals	Negative examples from three journals	Unlabeled examples from 99 journals
$rank \leq k$	21	18	561
$rank > k$	16	363	14,146

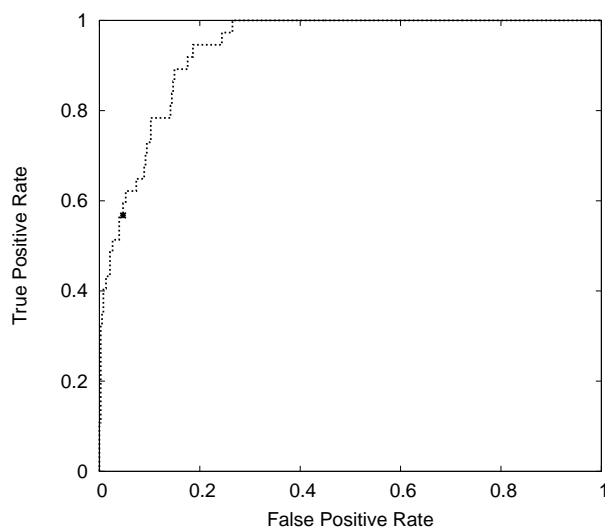


Fig. 2. An ROC curve showing the performance of our classifier on 418 articles from three journals, 37 of which were manually identified as relevant to TCDB. The remaining 381 articles are from the same journal issues and are considered negative examples. The point highlighted on the curve corresponds to a reasonable number of articles to show to the human expert.

Based on the threshold $k = 39$, our model recovers about 57% of the articles found in the literature. However, a much higher recall, up to 90%, is achievable with reasonable increase in human effort via a larger k . Moreover, 57% recall is acceptable because much relevant information is published more than once, and/or can be found by following citation paths starting at papers that are detected.

In the course of further research the human expert re-evaluated 25 of the articles from the 44 issues of the three journals that he dismissed previously. He then decided that 11 of these *are* relevant. This inconsistency indicates that the precision and recall of human procedures are far from perfect. Overall, the automated procedure has recall comparable to that of a human, and is definitely helpful in identifying articles that would otherwise be missed simply by browsing the literature. Indeed, at $k = 39$ our procedure recovers 8 of the additional 11 articles that the human found upon re-evaluation.

5 Deployment

We measure how effective our approach is in deployment by applying the classifier trained on the relabeled data set to a new data set. To do this, we download a new set of articles, again restricted to the same journals already referenced in TCDB, but now appearing in PubMed after the articles in the training set.

During these experiments, we generated five new data sets. These are produced when the human expert requests them, so they vary in size. For each set, we show the $k = 300$ highest-scoring articles to the human expert. After each set is deployed, the human expert screens and labels the top k . This means that we have access to an increasing number of training labels. After each deployment set is labeled, the classifier is retrained using the new labels.

Table 3. The labels given by the human expert to the top 300 articles identified by our trained classifier on five datasets spanning a period of approximately October 1, 2007 to March 13, 2007. The rightmost column indicates the proportion of the deployment sets which are true and false positives.

Approximately 4 weeks, October 1 - October 31 2007, 6108 articles

Label	Top 100	100-200	200-300	Total (%)
Accepted into TCDB	62	30	17	109 (1.78%)
Labeled relevant, but not accepted	12	12	12	36 (0.59%)
Not relevant	26	58	71	155 (2.53%)

Approximately 7 weeks, November 1 - December 20 2007, 10233 articles

Label	Top 100	100-200	200-300	Total (%)
Accepted into TCDB	54	33	9	96 (0.94%)
Labeled relevant, but not accepted	13	3	9	25 (0.24%)
Not relevant	33	64	82	179 (1.75%)

Approximately 3 weeks, December 21 - January 15 2008, 3885 articles

Label	Top 100	100-200	200-300	Total (%)
Accepted into TCDB	50	20	9	79 (2.03%)
Labeled relevant, but not accepted	28	21	10	57 (1.47%)
Not relevant	22	59	81	164 (4.22%)

Approximately 5 weeks, January 15 - February 20 2008, 6975 articles

Label	Top 100	100-200	200-300	Total (%)
Accepted into TCDB	51	27	8	86 (1.23%)
Labeled relevant, but not accepted	28	39	20	87 (1.25%)
Not relevant	21	34	72	127 (1.82%)

Approximately 3 weeks, February 21 - March 13 2008, 2544 articles

Label	Top 100	100-200	200-300	Total (%)
Accepted into TCDB	24	13	6	43 (1.69%)
Labeled relevant, but not accepted	38	19	8	65 (2.56%)
Not relevant	38	68	86	192 (7.55%)

Table 3 shows the disposition of these top 300 articles, as assigned by the expert, in each of the five deployment sets. As expected, the precision of our models decreases with the scoring rank of the article: the top 100 articles are

more likely to be genuinely relevant than the articles ranked 100-200 or 200-300. The proportion of relevant documents varies from 1.18% (deployment set 2, November and December 2007) to 4.25% (deployment set 5). It is not obvious what the source of this variability is. The source may be underlying variability in the content of the journals, and it may be variation in the standards of the human expert, who is influenced by the overall quality and quantity of the deployment sets. Note that the proportion of articles accepted into TCDB varies less, ranging between 0.94% (deployment set 2) and 2.03% (deployment set 3), indicating that our approach is able to provide a consistent stream of important documents.

6 Conclusion

We have shown four results that we believe are interesting and of general significance. First, SVMs can be used effectively as a method of ranking text documents according to likely relevance to a specialized biomedical database, and not just as a method of yes/no classification. Second, our method for iteratively relabeling examples is an efficient and effective approach to the positive-only issue. Third, this issue is in fact less important than it appears, because of the equivalence established in Lemma 1. Fourth, our methods can successfully provide an up-to-date stream of highly relevant documents to the human maintainers of TCDB, a widely-used protein database. In total, these methods have so far discovered 964 relevant articles, 626 of which have been added to TCDB. This represents a 16% increase in the number of references in TCDB over the last six months, which is much faster than the previous rate of growth of TCDB.³

Now that we have established that our approach is useful for updating real databases like TCDB, we plan to apply our learned classifiers to papers published outside the set of journals already known to contain some relevant articles, in order to find relevant articles that would otherwise certainly be missed. We also plan to extend our feature set. In particular, Wang *et al.* [16] show that author names, medical subject headings (MeSH) and standardization of biological strings (*e.g.* converting “9-mer” to “x-mer”) can improve performance. Han *et al.* [7] hypothesize that a characteristic of biomedical text is that it contains informative suffixes and show that using substrings as features can outperform general-purpose stemming algorithms, such as the Porter stemmer, when classifying biomedical text. In addition, we plan to incorporate journal identity, title words, paper length, paper category (*e.g.* based on keywords or as curated by Medline), and paper type (*e.g.* review) as document features.

Recall that *Nucleic Acids Research* lists over 1,000 specialized databases. Although our experiments to date focus exclusively on only one of them, our methods are directly applicable to many more of these databases.

Acknowledgments. This research is funded by NIH grant GM077402.

³ Most of these articles were discovered during the five deployment sets described in Section 5, but some came from the iterative relabeling experiments described in Section 3 and some came from various experimental trials not described above.

References

1. S. Das, M. H. Saier Jr., and C. Elkan. Finding transport proteins in a general protein database. In *Proceedings of the Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 54–66, 2007.
2. F. Denis, R. Gilleron, and F. Letouzey. Learning from positive and unlabeled examples. *Theoretical Computer Science*, 348(1):70–83, 2005.
3. P.B. Dobrokhotov, C. Goutte, A. L. Veuthey, and E. Gaussier. Combining NLP and probabilistic categorisation for document and term selection for Swiss-Prot. In *Proceedings of the Eleventh International Conference on Intelligent Systems for Molecular Biology*, pages 91–94, 2003.
4. P.B. Dobrokhotov, C. Goutte, A. L. Veuthey, and E. Gaussier. Assisting medical annotation in Swiss-Prot using statistical classifiers. *International Journal of Medical Informatics*, 74(2-4):317–24, 2005.
5. C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008)*, pages 213–220, 2008.
6. M.Y. Galperin. The Molecular Biology Database Collection: 2008 update. *Nucleic Acids Research*, 36(Database issue):D2, 2008.
7. B. Han, Z. Obradovic, Z. Hu, C. H. Wu, and S. Vucetic. Substring selection for biomedical document classification. *Bioinformatics*, 22(17):2136–2142, 2006.
8. T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.
9. T. Joachims. A support vector method for multivariate performance measures. *ACM International Conference Proceeding Series*, 119:377–384, 2005.
10. M. H. Saier Jr. A functional-phylogenetic classification system for transmembrane solute transporters. *Microbiology and Molecular Biology Reviews*, 64(2):354–411, 2000.
11. M. H. Saier Jr., C. V. Tran, and R. D. Barabote. TCDB: The transporter classification database for membrane transport protein analyses and information. *Nucleic Acids Research*, 34:D181–D186, 2006.
12. B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. Building text classifiers using positive and unlabeled examples. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003)*, pages 179–188, 2003.
13. A. K. McCallum. MALLET: A machine learning for language toolkit. <http://mallet.cs.umass.edu>, 2002.
14. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
15. C. Wang, C. Ding, R. F. Meraz, and S. R. Holbrook. PSoL: A positive sample only learning algorithm for finding non-coding RNA genes. *Bioinformatics*, 22(21):2590–2596, 2006.
16. P. Wang, A. A. Morgan, Q. Zhang, A. Sette, and B. Peters. Automating document classification for the immune epitope database. *BMC Bioinformatics*, 8(269), 2007.
17. W. J. Wilbur. Boosting naïve Bayesian learning on a large subset of MEDLINE. In *Proc. AMIA Symp.*, 2000.