Boosting the Feature Space: Text Classification for Unstructured Data on the Web

Yang Song[1], Ding Zhou[1], Jian Huang[2], Isaac G. Councill[2],
Hongyuan Zha[1,2], C. Lee Giles[1,2]
[1]Department of Computer Science and Engineering,
[2]College of Information Sciences and Technology,
The Pennsylvania State University,
University Park, PA 16802, U.S.A.

Abstract

The issue of seeking efficient and effective methods for classifying unstructured text in large document corpora has received much attention in recent years. Traditional document representation like bag-of-words encodes documents as feature vectors, which usually leads to sparse feature spaces with large dimensionality, thus making it hard to achieve high classification accuracies. This paper addresses the problem of classifying unstructured documents on the Web. A classification approach is proposed that utilizes traditional feature reduction techniques along with a collaborative filtering method for augmenting document feature spaces. The method produces feature spaces with an order of magnitude less features compared with a baseline bag-of-words feature selection method. Experiments on both real-world data and benchmark corpus indicate that our approach improves classification accuracy over the traditional methods for both Support Vector Machines and AdaBoost classifiers.

1 Introduction

With the tremendous growth of the World Wide Web, content classification [3, 9] has become an efficient approach to organize the contents of large corpora, as well as providing enhanced search features and recommendations. However, experience with the CiteSeer Digital Library[1] indicates that there exist several challenges in text classification for unstructured data on the Web, particularly when the number of classification labels is large.

First, since CiteSeer [1](and most search engines) automatically crawls academic documents from venue websites, author homepages and then extracts textual information from them to create metadata, false labels are inevitably assigned to many documents. Due to the increasing similarities between different venues (e.g., *SIGKDD* and *PKDD*, *ECML* and *ICML*), the effort needed to accurately classify a document into exactly one category becomes greater. Moreover, lack of keyword fields and other feature deficiencies create unique challenges for text classification.

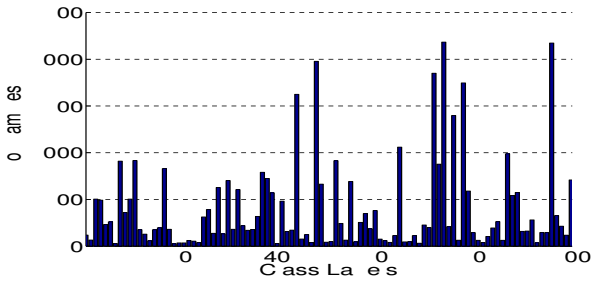


Figure 1. Distribution of documents w.r.t. classes in CiteSeer. In Practice, documents on the Web are also unevenly distributed.

This problem is further exacerbated due to the imbalance of documents available for training in each class, i.e., the documents are unevenly distributed in different categories on the Web (for example, CiteSeer has a collection of more than 3,000 documents for *INFOCOM*, while for some other conferences, the cumulative numbers are no more than 200); Figure 1 gives an actual document distribution on one of the data sets used later in the paper.

The major contributions of this work are (1) an evaluation of the use of collaborative filtering for refining minimal, noisy feature spaces, and (2) a comparison of the performance of SVM versus AdaBoost classifiers for the

[1]http://citeseer.ist.psu.edu/

Proceedings of the Sixth International Conference on Data Mining (ICDM'06)
0-7695-2701-9/06 $20.00 © 2006 IEEE

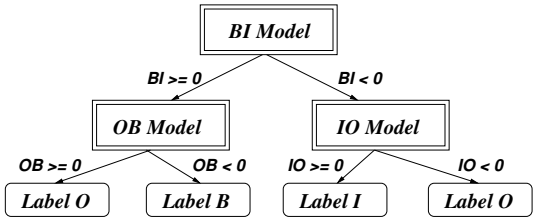


Figure 2. Two-level decision tree for tagging.

problem of categorizing academic documents by publication venue. Collaborative filtering is employed to predict the value of missing features for each class. Experimental evaluations on both real-world data set and benchmark corpus show great improvement with regard to classification accuracy compared with classification using the original feature space and the feature selection method — Information Gain (IG).

2 Entity Extraction using SVM-decision-tree

To identify non-trivial noun phrases with semantic meanings in the documents, noun phrases (NP) chunking is adopted for this purpose. Chunking groups together semantically related words into constituents, a step beyond POS (Part-Of-Speech) tagging; but it does not identify the role of these constituents with respect to the sentence, nor their inter-relationships. In our system, we revised and implemented a previous chunking algorithm [7] as a simplified yet more efficient two-level SVM-based NP chunker.

The NP chunking problem is formalized as a three class classification problem, which assigns each word with one of the labels: B (Beginning of NP), I (Inside NP), O (Outside NP). A feature space is constructed, with dimensions representing the surrounding words, the POS tags of those words, and the already tagged chunk tags. Three SVM models (BI, IO, OB) are trained, each designed to tag a word in favor of one label over the other, for example, the BI model provides a hyperplane to tag a word with label B rather than label I.

We adopt the pairwise method that allows the SVM to classify multi-class problems. Traditional methods have considered using three SVMs together at each time, i.e., in the worst case, three comparisons need to be made in order to determine the label of a word. However, we use a method that allows us to use two SVMs instead of three, which in turn accelerates the chunking time by one third. The hierarchy of the two-level decision tree employed is shown in Figure 2. Furthermore, Table 1 shows an example to clarify the method we use.

Given a paragraph of unstructured text, the extraction goes through the steps of sentence segmentation, POS tagging using Brill tagger[2] and NP chunking. In this example, *collaborative* and *filtering* are labeled as adjective and noun by Brill tagger, the chunking decision for *collaborative* is based on the results of the SVMs: the result of BI model is 0.5 (in favor of label B), so the OB model is used which yields -0.6 (in favor of label B), thus B-NP is chosen as the chunk tag for this word.

current word	POS tag	chunk tag
This	DT	B-NP
paper	NN	I-NP
describes	VBZ	O
our	PRP	B-NP
attempt	NN	I-NP
to	TO	O
unify	VB	O
entity	NN	B-NP
extraction	NN	I-NP
and	CC	O
collaborative	JJ	B-NP
filtering	NN	I-NP
to	TO	O
boost	VBG	O
the	DT	B-NP
feature	NN	I-NP
space	NN	I-NP
.	.	O

Table 1. Chunk representation example. Each word is first tagged with POS tag, and POS tags are then classified into B-NP, I-NP and O tags.

3 Feature Space Refinement

Inspired by the analogy between *user behaviors* and *venue focuses* (i.e., different users may have similar preferences, different conferences/journals may focus on the same research areas), we employ collaborative filtering (CF) to refine the feature space by predicting missing values as well as reducing noise factors from the feature space.

3.1 Algorithm and Analysis

Considering the large number of examples we collected, it is impractical to apply traditional memory-based CF approaches. Instead, we use instance selection techniques to choose small portions of *candidates* each time for prediction. In our proposed Algorithm 1, we set a threshold and dynamically choose $\log m$ candidates according to

[2] http://research.microsoft.com/∼Ebrill/

the weight function (see below), where m denotes the total number of examples we collected.

The feature space F is normalized before applying to the algorithm. Unlike traditional applications, the value in the feature space does not have a unified ratings scale, i.e., each feature may have different frequencies for different examples and does not have a uniform upper-bound. The feature space is normalized as follows:

$$||F_{ij}||_2 = \frac{||F_{ij}||_2}{\max_{1 \le j \le n} ||F_{ij}||_2} \quad (1)$$

where $\|\cdot\|_2$ means the Euclidean norm. The weight matrix determines the similarity between examples, it is crucial for setting up a criteria of choosing the right candidates in Algorithm 1. We compute the matrix $\mathcal{W} \in \mathbb{R}^{m \times m}$ as follows:

$$\mathcal{W}(a,i) = \frac{\sum_{j=1}^{n} \mathcal{H}(F_{aj}, \epsilon_a)\mathcal{H}(F_{ij}, \epsilon_i) Sim(a,i,j)}{min\,(I_a, I_j)} \quad (2)$$

where $\mathcal{H}(.,.)$ is a binary classification function:

$$\mathcal{H}(a,b) = \begin{cases} 1 & \text{if } a > b; \\ 0 & \text{otherwise.} \end{cases}$$

The $Sim(.,.,.)$ function calculates the closeness between example a and example i over feature j, with S as the frequency scale. Defined as:

$$Sim(a,i,j) = \frac{(S - |F_{aj} - F_{ij}|)}{S} \quad (3)$$

The vector $\epsilon = \{\epsilon_1, \epsilon_2, ...\epsilon_n\}$ are some significant small coefficients that are a little bit less than the lowest feature frequencies for each example. Specifically,

$$\epsilon(F) = \min_{i \neq 0} \frac{||F_i||_1}{||i||_1}, \quad (4)$$

where $\|\cdot\|_1$ is the taxicab norm.

After computing the weight matrix $\mathcal{W}$, Algorithm 1 is called to retrieve the candidates list for each active user a. In Algorithm 1, ξ is a lower bound for selecting the candidates based on weight similarity. Notice that ξ is the only parameter we need to tune for the algorithm, which directly affects the efficiency and outcome of the algorithm. If the chosen ξ is too low, the algorithm may not be able to retrieve enough candidates that satisfy the condition on line 4; on the other hand, if using a threshold that is too high, it may introduce too much overhead for sorting the candidates array that dominates the running time for the algorithm. By choosing a threshold through cross-validation, Algorithm 1 only needs $O(m)$ time to select candidates for each example, i.e., the candidates list can be learned in one pass through the data. Afterwards, the predicted frequency of feature j from example a can be calculated as follows:

Algorithm 1 Candidates Selection

procedure Candidate-Selection
input: example-feature matrix $F \in \mathbb{R}^{m \times n}$
weight matrix $\mathcal{W} \in \mathbb{R}^{m \times m}$
output: candidates list $\mathcal{C} \in \mathbb{R}^{\lceil \log m \rceil \times n}$
1. Initialize $\mathcal{C} \leftarrow \emptyset, \eta \leftarrow 0, \tau \leftarrow \xi$
2. **for** each target example a **do**
3. **for** i equals 1 to m **do**
4. **if** $\mathcal{W}(a,i) > \tau$
5. $\mathcal{C}_a \leftarrow \mathcal{C}_a \cup F_i$ [update the candidate list for a]
6. $\eta \leftarrow \eta + 1$ [η monitors the length of $\mathcal{C}$]
7. **if** $\eta > \lceil \log m \rceil$
8. Sort $\mathcal{C}_a$ in descending order of $\mathcal{W}$
9. $\mathcal{C}_a \leftarrow \mathcal{C}_a - \mathcal{C}_{\lceil \log m \rceil}$ [remove the last element]
10. $\eta \leftarrow \eta - 1$
11. **end**
12. **end**
13. output $\mathcal{C}$

$$\mathcal{P}(a,j) = \sum_{i=1}^{\lceil \log m \rceil} \mathcal{W}(a,i) * \mathcal{C}_{ij} \quad (5)$$

where $\mathcal{C}_{ij}$ means the rating of ith candidate over item j. Note that the computational cost for prediction is significantly slashed to $O(n \log m)$ for each example. In practice, we find out that the *MAE*[3] scores of equation (5) are very low, which mainly attributes to the good selection result by algorithm 1. The candidates selected are almost examples that come from the same class as the example the prediction is for, which guarantees optimal prediction results.

4 Experiments and Discussions

In the experimental evaluation, we ran a series of experiments to compare our proposed approach with traditional methods on two data sets: CiteSeer Digital Library and WebKB benchmark corpus[6]. Specifically, three kinds of experiments are carried out:

First, we make comparison between entity extraction techniques in terms of the dimensionality of the feature space. We compare our proposed *SVM-decision-tree* approach to the *bag-of-words* method with the standard TFIDF approach as an extension. To be more convincing, *Information Gain* (IG) is applied to the *bag-of-words* approach as a feature selection criteria. A feature y is deemed

[3] *MAE* (Mean Absolute Error) represents how much the mean predicted values deviate from the observed values of all examples in the data set. $MAE_a = \frac{1}{m_a} \sum_{j \in P_a} |p_{a,j} - o_{a,j}|$. Obviously, the lower MAE is, the better the prediction is.

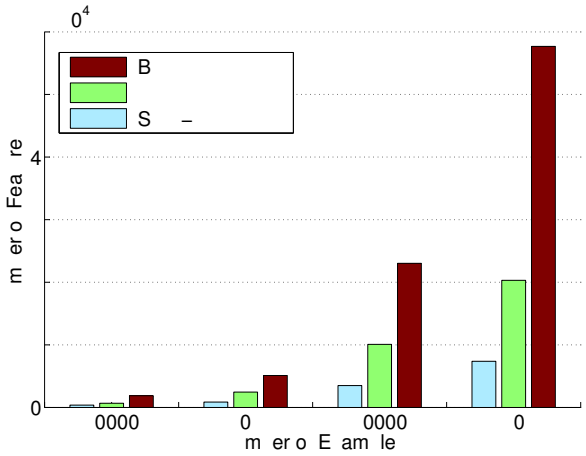


Figure 3. Features extracted by *bag-of-words* (BOW) and *SVM-decision-tree* (S-D) from the summarizing parts of the documents in CiteSeer data set, where S-D creates a much smaller feature space as a function of example size. The number of examples chosen by IG is decided by maximizing the F-measure on the validation set.

useful if its expected IG exceeds the threshold.[4] Comparisons are also made between IG and *SVM-decision-tree*.

Furthermore, to illustrate that the CF algorithm indeed boosts the feature space, we compare the distribution of features in each class Before Collaborative Filtering (B-CF) and After Collaborative Filtering (A-CF). To be more convincing, we also calculate the distribution of features from prediction results by using the classic Inner Product approach (I-CF)[5] proposed by Breese et al. in [2].

Finally, we use multiclass SVM [5] and AdaBoost.MH [4] to classify the feature space extracted by (1) not using CF (B-CF-SVM and B-CF-Boost), (2) using IG feature selection (IG-SVM and IG-Boost), and (3) using CF (A-CF-SVM and A-CF-Boost). The Vector Similarity method (VSIM) is used as baseline for comparison. Additionally, since it was shown that SVMs can perform well even without feature selection [8](SVM-No), it is also compared in the experiment. We apply *Precision*, *Recall* and *F-measure* as measures for our text classification.

[4]Experimentally, the threshold is usually chosen to maximize the F-measure on a validation set.

[5]The weight function is calculated as $w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k\in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k\in I_i} v_{i,k}^2}}$, and the prediction score is computed through the whole data set.

4.1 CiteSeer Data Preparation

CiteSeer is one of the largest digital libraries that holds currently more than 740,000 documents. As mentioned in Section 2, we only consider extracting entities from the summarizing parts of the documents, i.e., the titles, abstracts and keyword fields. Document class labels are obtained from the *venue impact page*[6]. We use only the top 200 publication venues listed in DBLP in terms of impact rates, each of which was referred as a class label. Overall, the total number of documents we used for the experiments is up to 118,058, divided into training set and testing set by doing 10-fold cross-validation. Notably, we keep the imbalance of the classes.

4.2 Experimental Results on CiteSeer Data Set

Figure 3 presents the number of features extracted by the three techniques. We ran the experiments with the number of documents, D, equal to 10,000, 25,031, 50,000 and 118,058. Using *SVM-decision-tree* approach yields a much lower-dimensional feature space compared with the *bag-of-words* method (with TFIDF), especially when the number of examples are very large. Information Gain successfully reduces the feature space to half the dimension of *bag-of-words*, but when the training data size becomes larger (118,058), it still creates a feature space of more than 20,000 features, while our approach ends up with a feature space with a little more than 7,000 features. We also notice that the dimension of feature space generated by our approach is almost linear to the number of examples, indicating nice scalability of our entity extraction technique.

In Figure 4, we depict the distribution of features for three approaches that applied to the feature space extracted by *SVM-decision-tree* approach. Before applying CF algorithm (B-CF), the features are unevenly distributed in each class due to the random distribution of training examples in different categories. By using the Inner Product algorithm (I-CF) it first computes the correlations between each pair of examples, and then predicts the feature frequencies from the knowledge of all examples. As a result, I-CF generates too many features for each class that inevitably causes overlapping in the feature spaces, which leads to reduction of classification accuracy. Finally, by employing the CF algorithm we proposed (A-CF), the feature space is boosted to a reasonably dense level that yields a nearly even distribution of features in each class. The virtue of the boosted feature space is not only that it contains enough features within each class which makes it easy to classify, but also results in very little overlapping of different classes in the feature space, which reduces the misclassification rate significantly in comparison with I-CF. Figure 5 compares the

[6]http://citeseer.ist.psu.edu/impact.html

	examples	VSIM	SVM-No	B-CF-SVM	B-CF-Boost	IG-SVM	IG-Boost	A-CF-SVM	A-CF-Boost
Precision	10,000	24.31	62.17	46.44	65.74	53.77	56.29	80.25	**85.24**
	25,031	25.17	85.77	68.33	82.33	85.63	87.21	91.01	**94.08**
	50,000	25.24	86.02	70.47	82.53	86.31	88.24	92.53	**93.22**
	118,058	27.96	89.42	82.77	85.32	89.32	89.33	**95.77**	94.66
Recall	10,000	10.23	13.75	11.43	11.77	11.85	12.11	**14.53**	12.11
	25,031	25.72	33.23	26.22	30.25	28.53	31.74	**42.77**	40.69
	50,000	34.81	**50.25**	35.79	29.88	42.79	40.01	**50.25**	49.00
	118,058	72.38	84.88	74.25	77.91	83.99	72.53	**85.27**	73.00
F-measure	10,000	14.40	22.52	18.34	17.73	20.11	18.35	24.61	**25.21**
	25,031	25.44	47.90	37.89	44.24	38.29	48.32	**58.19**	56.81
	50,000	29.26	63.25	47.47	43.88	56.77	60.11	64.13	**64.24**
	118,058	40.34	87.09	78.28	81.45	79.52	83.23	**90.22**	81.14

Table 2. Experimental results of CiteSeer data set in terms of Precision (%), Recall(%) and F-measure(%), averaged over all classes. VSIM is compared as a baseline approach. Our approach (A-CF) shows competitive results on both classifiers. IG chooses top k features to maximize the F-measure of the validation set. For the entire data set (118,058), k is around 20,000.

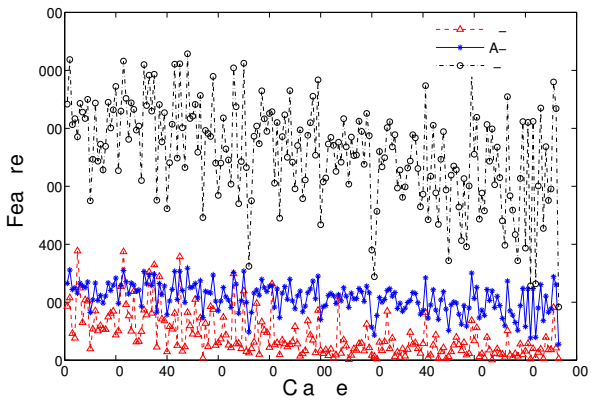


Figure 4. Feature distribution where B-CF denotes the feature space before applying CF, I-CF the feature space augmented by Inner-Product method, and A-CF the feature space augmented by our CF algorithm.

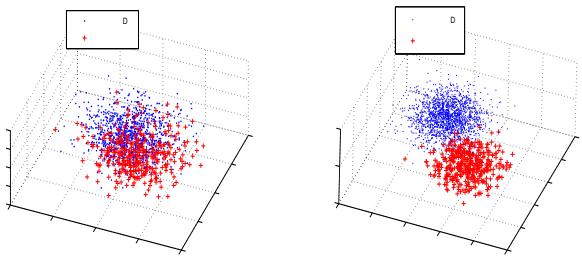


Figure 5. Visualization of SVD feature distribution of classes (*SIGMOD, WWW*). The left figure shows features by the I-CF inner-product, the right figure the boosted feature space by our A-CF algorithm.

feature spaces for 2 classes by applying I-CF and A-CF, respectively. We use *Singular Value Decomposition* (SVD) to get the first 3 *principal components* of the matrix and visualize in a 3-D graph. It is not hard to see that I-CF leads to a much more overlapping space than our approach, which generally separates two classes very well.

Table 2 summarizes experimental results for the three metrics averaged over all classes. With regard to *precision*, our approach achieves significant improvement on both classifiers. When the number of examples is small (10,000), A-CF-SVM and A-CF-Boost improve the precision over the VSIM baseline approach by nearly 4 times, and nearly twice as much as the results of Information Gain (IG-SVM and IG-Boost). When the whole data set is applied to the experiment (118,058), A-CF-SVM and A-CF-Boost achieve the best results of 95.77% and 94.66% respectively, about 5% more than IG. Meanwhile, without feature selection, SVM (SVM-No) shows almost the same precision as IG-SVM, with a slightly better result when the number of examples is small.

In terms of *recall*, all methods have very close performances. Comparatively, SVM performs slightly better than AdaBoost regardless of data size and entity extraction techniques. Especially when the data size is large (118,058),

No. of pages	*BOW+TFIDF*	*IG*	*SVM-DT*
1,000	2,413	1,533	**977**
2,000	4,987	2,422	**1,777**
4,000	7,400	5,324	**3,599**
8,282	10,322	6,891	**5,111**

Table 3. Number of features exacted by three techniques w.r.t. number of pages for the WebKB data set. SVM-DT approach yields a much smaller feature space.

the baseline approach achieves recall of 72.38 %, almost the same as A-CF Boost method (73.00%). However, both of which are nearly 13% lower than A-CF-SVM approach. Both SVM-No and A-CF-SVM achieve the best recall among all when the number of examples equals 50,000.

Our approach outperforms IG for both classifiers in terms of *F-measure*, with an exception when the data size is 118,058, IG-Boost outperforms A-CF-Boost by 2%. Both IG-Boost and A-CF-Boost are almost 10% less than that of A-CF-SVM method, which shows the best performance of all. During the experiments, we also noticed that the training time of SVM and AdaBoost are almost the same with SVM slightly better in some cases.

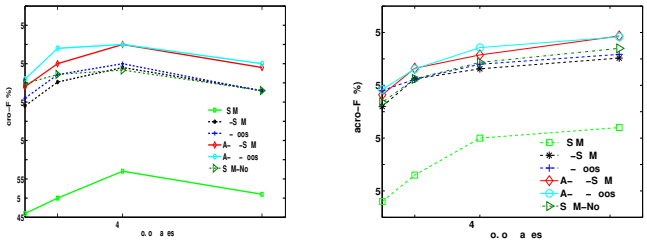
4.3 WebKB:World Wide Knowledge Base

The WebKB data set contains web pages collected from cs departments of many universities by the World Wide Knowledge Base project of the CMU text learning group in January 1997. For performance evaluation, we divide the data into training and testing set with the proportion of 4:1. A series of experiments were performed with the number of documents equal to 1,000, 2,000, 4,000 and 8,282. The number of iterations T for AdaBoost is set to 500.

Table 3 summarizes the number of features with regard to the training data size. The *SVM-decision-tree* approach creates a much smaller feature space than *bag-of-words* and *Information Gain*—20% less than the IG and 50% less than the BOW when the total WebKB collection is used.

Figure 6(a) shows the result of the *Micro-F* scores. When the number of training pages is small, our approach has almost the same performance as IG for both classifiers, with less than 2% improvement. As the page size get larger, the performance improvement of our approach becomes greater. When the whole collection is used, our approach outperforms IG by more than 5%, but the performance decreases for both methods as the best results are achieved when the page size is 4,000. Note that SVM-No has almost the same performance as IG-SVM.

Macro-F scores are shown in Figure 6(b). Clearly, the



(a) (b)

Figure 6. Micro-F(a) and Macro-F(b) results for WebKB w.r.t. data size.

baseline approach VSIM performs the worst regardless of data size. SVM-No again performs nearly the same as IG-SVM. Note that with the increase of pages, the macro-F scores increase as well for all methods. Our approach generally outperforms IG, and the advantage becomes larger with the increase of data size. Our approach achieves a significant improvement by 8% over IG for both classifiers when the whole WebKB collection is applied.

References

[1] K. Bollacker, S. Lawrence, and C. L. Giles. CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proceedings of AGENTS '98*, pages 116–123, New York, 1998. ACM Press.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Uncertainty in Artificial Intelligence. Proceedings of the Fourteenth Conference (1998)*, pages 43–52.

[3] L. Cai and T. Hofmann. Text categorization by boosting automatically extracted concepts. In *SIGIR '03*, pages 182–189, New York, NY, USA, 2003. ACM Press.

[4] M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. *Mach. Learn.*, 48(1-3):253–285, 2002.

[5] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, 2002.

[6] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *AAAI '98/IAAI '98*, pages 509–516, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

[7] T. Kudo and Y. Matsumoto. Use of support vector learning for chunk identification. In *Proceedings of the 4th Conference on CoNLL-2000 and LLL-2000*, pages 142–144, 2000.

[8] H. Taira and M. Haruno. Feature selection in svm text categorization. In *AAAI '99/IAAI '99*, pages 480–486, Menlo Park, CA, USA, 1999.

[9] D. Zhuang, B. Zhang, Q. Yang, J. Yan, Z. Chen, and Y. Chen. Efficient text classification by weighted proximal svm. In *ICDM*, pages 538–545, 2005.