Email Classification with Co-Training

Svetlana Kiritchenko and **Stan Matwin**

School of Information Technology and Engineering
University of Ottawa
Ottawa, ON, Canada
{svkir,stan}@site.uottawa.ca

Abstract

The main problems in text classification are lack of labeled data, as well as the cost of labeling the unlabeled data. We address these problems by exploring co-training - an algorithm that uses unlabeled data along with a few labeled examples to boost the performance of a classifier. We experiment with co-training on the email domain. Our results show that the performance of co-training depends on the learning algorithm it uses. In particular, Support Vector Machines significantly outperforms Naive Bayes on email classification.

1 Introduction

You have just returned from a relaxing two-week vacation. There has been no phone, no email for two wonderful weeks, and now you are back. You open your inbox and ... Wow! There are 347 new messages! How could you manage to read all of them? Probably, you will spend the whole day trying to sort out all this mail. Having done this burdensome work, you feel like you need a vacation again. What is worse is that most of those messages are out of your interest or out of date.

Nowadays, a typical user receives about 40-50 email messages every day. For some people hundreds of messages are usual. Thus, users spend a significant part of their working time on processing email. As the popularity of this mean of communication is growing, the time spent on reading and answering emails will only increase. At the same time, a large part of email traffic consists of non-personal, non time-critical information that should be filtered. As a result, there has recently been a growing interest in creating automatic systems to help users managing an extensive email flow. Examples of such systems are PEA[19], MailCat[16], Re:Agent[2] and others.

Generally, the main tool for email management is text classification[15, 4, 3]. A classifier is a system that automatically classifies texts into one (or more) of a discrete set of predefined categories. For example, for email management one could benefit from a system that classifies incoming messages as junk and non-junk or as important and unimportant.

Most text classification approaches use supervised learning for building a classification system. In a supervised learning setting, we are given examples that belong to a two-class concept (e.g. interesting and uninteresting email). All examples are labeled with respect to their membership in one of the two classes. A machine learning system induces, from these examples (referred to as a training set), a general description of both classes. It is important that such a description has predictive power, i.e. it predicts with high degree of success the class of unseen examples. In our example, we want a machine learning system to come up with a definition of what makes an interesting email message, so that this definition will successfully pick interesting email from the user's inbox in the future (e.g. it will work well on examples different from the ones in the training set).

It is well-known, both theoretically[17] and practically[4, 9], that more training data we have, the more accurate classification system we get. In general, we need hundreds or even thousands of labeled examples to produce a reasonably accurate classifier. For example, recently Microsoft has released the product Outlook Mobile Manager that can prioritize users' incoming messages and send the most important ones (along with other information like current tasks, reminders and so on) to their mobile devices. Microsoft claims that the system would work its best after getting about 600 labeled examples. For an average user it would require several weeks for the system to start working in full. Moreover, during these weeks the user would be constantly bothered to label each incoming message as important or not.

One way to overcome the problem of laborious hand-labeling is to look at the user's behavior to determine if a message is important or not. For example, if a message has been printed, it is probably an important one. On the other hand, deleting a message without reading could be the indication of junk. But this kind of monitoring is not very reliable and also requires some time to gather information before the system is able to learn the generalization.

Another way to deal with the problem of labeling data is putting in use unlabeled data that we usually have plenty of. There have been proposed some learning algorithms that are designed for the extensive use of unlabeled data. In this paper we will concentrate on one such algorithm called Co-Training[1]. This algorithm allows one to start with just a few labeled examples to produce an initial weak classifier and then use only unlabeled data to improve the performance. It is based on the idea that sometimes features that describe the data are redundant and could be split into two sets each of which on its own is sufficient for correct classification. Then we can build two classifiers, one for each set. We use these two classifiers to go through unlabeled examples, label them, and add the most confident predictions to the labeled set. In other words, the classifiers train each other using the unlabeled data[1]. Craven et al. employed co-training in their large research project on extracting knowledge from the World Wide Web[5]. The results were very encouraging; they were able to reduce the classification error by better than a factor of two using only unlabeled data.

The goal of this research is exploring the possibility of applying co-training to email classification. Usually, in email classification (as well as in text classification in general) texts are represented as bags of the words which appear in a whole message, i.e. in the header and in the body of a message. But often the information in a header and in a body is similar. In other words, only one of the sources, either a header or a body, is sufficient for correct classification. In [3], the authors empirically show that using only headers just slightly reduces the accuracy of classification. Therefore, we split the bags of words that represent email messages into two sets: the words from headers[2] and the words from bodies. We then proceed with the co-training algorithm as described in [1]. Our results show that the performance of co-training depends on the learning algorithm it uses. In particular, Naive Bayes[11] could not benefit from co-training on our data whereas Support Vector Machines(SVM)[18] has been applied successfully. The possible explanation to this phenomenon is the sparseness of feature vectors that SVM could deal with efficiently and that is crucial for Naive Bayes.

The contribution of this work is two-fold: firstly, we show in detail how to apply co-training on text-mining tasks. Secondly, we demonstrate that SVM is the learner of choice in these applications of co-training.

In the remainder of this paper, we first give a short survey of related work, then describe the co-training algorithm in detail. After that, we present our results of applying the algorithm to

[1]It is interesting to observe that such a learning setting involving two (or more) agents reflects the social character of the learning process, unlike the standard Machine Learning approach that focuses on an isolated, single learner.

[2]In this study we use only a subject instead of a whole header for reasons mentioned in Section 4.

email classification. Finally, we give our conclusions and discuss the prospects for future work.

2 Related Work

After appearing of the original work [1], a number of studies emerge to explore the potentials of the co-training idea. For example, in [14] co-training has been successfully applied to one of the natural language processing tasks, namely base noun phrase bracketing, which usually requires a large amount of training data. In this work the authors also showed that a significant improvement in accuracy can be achieved by combining co-training and active learning in the form of using humans to correct inaccurate labels made by co-training.

Another variant of combining co-training and active learning has been proposed in [12]. Co-Test(Co-EM) extends co-training in that it asks to explicitly label examples on which the two classifiers of co-training have different opinions.

Nigam and Ghani[13] performed extensive experiments comparing the performance of co-training and another popular algorithm that uses unlabeled data: Expectation-Maximization(EM)[6]. These experiments show that co-training outperforms EM even on tasks where there is no natural split of features.

Transductive Support Vector Machines(TSVM) proposed in [9] is another semi-supervised learner that to some extent subsumes co-training[10]. It uses labeled and unlabeled data to find the maximum margin hyper-plane dividing the positive and negative instances. It is particularly beneficial in the situations where we do not care about good generalization, but rather good classification accuracy on a particular test set.

In addition, [21] has presented a method of using a large amount of unlabeled information available on the World Wide Web to improve the classification of short text strings.

3 Co-Training Algorithm

In this section we give a detailed description of the co-training algorithm adopted from [1].

Sometimes features describing the data are redundant for a given task, so that we can classify an example having only one set of features or another. Such sets of features are called "redundantly sufficient". For example, emails can be classified using only one of the two sets of features: header information (for example, subjects) or the words in the bodies of messages. We often can say what a message is about looking at the message itself or looking only at the subject. We split features into two sets and train two independent classifiers, one for each set of features. We train them providing some minimal number of labeled examples that we have and get two weak classifiers that are, hopefully, better then random. Then, we use unlabeled data in a loop for some number of iterations or as long as we have data. In real settings, one can find the optimal number of iterations testing the performance on each iteration on a separate validation set. In the loop, both classifiers examine new examples, label them, and add the most confidently predicted positive and negative examples to the set of labeled examples. Thus, the classifiers train each other using only unlabeled data.

The whole algorithm is as follows:

Given:

- F_1 and F_2 are redundantly sufficient sets of features
- $\mathbf{L}$ is a set of labeled training examples
- $\mathbf{U}$ is a set of unlabeled examples

Loop:

- Learn the first classifier $\mathbf{C}_1$ from $\mathbf{L}$ based on $\mathbf{F}_1$
- Learn the second classifier $\mathbf{C}_2$ from $\mathbf{L}$ based on $\mathbf{F}_2$
- For each classifier $\mathbf{C}_i$ do:
    - $\mathbf{C}_i$ labels examples from $\mathbf{U}$ based on $\mathbf{F}_i$
    - $\mathbf{C}_i$ chooses p positive and n negative the most confidently predicted examples E from $\mathbf{U}$
    - $\mathbf{C}_i$ removes examples E from $\mathbf{U}$
    - $\mathbf{C}_i$ adds examples E with the corresponding labels to $\mathbf{L}$

Why should this work? The intuition is that if one classifier can find an example that is very similar to some of labeled ones, then it can confidently predict the class of this example and provide one more training example for the other classifier. But, of course, if this example happened to be easy to classify for the first classifier, it does not mean that this example will be easy to classify for the second classifier, so the second classifier will get useful information to improve itself and vice versa. For example, if we have two messages, one with subject "Company meeting today at 3 pm" and the other one with subject "Meeting today at 5 pm?", and we know that the first message is classified as "meeting", then we are very confident that the second message should be classified as "meeting" as well based only on its subject. But the messages are likely to have different content, especially, if they are from different people. Therefore, the classifier that is based on the words in the body of a message will be provided with a whole bunch of new words that are relevant to class "meeting".

Blum and Mitchell ran some experiments to see if co-training really worked. They considered the problem of classifying universities' web pages as home pages of academic courses. The two sets of features were the words on pages themselves and the words on the hyperlinks that point to the page. The underlying learning scheme for co-training was Naive Bayes. They provided just 12 labeled examples (3 positive and 9 negative) and approximately 800 unlabeled ones. On each iteration of co-training, each classifier was allowed to add 1 new positive and 3 new negative examples to the pool of labeled examples. After 30 iterations of the co-training algorithm, the accuracy of the classifier increased reducing the error by better than a factor of two.

In [1] the authors also provided some theoretical insight into the co-training method. They proved that if the feature sets representing the data are conditionally independent given the class and the target classification function is learnable[3], then any initial weak classifier can be boosted to arbitrarily high accuracy using unlabeled examples only.

[3]We use "learnable" in terms of the general learnability model (PAC model) that assumes that the more examples are in the training set, the better the accuracy on the testing set is, provided that the distribution of examples in both sets is the same[17]

4 Co-Training on Email Classification

Inspired by the success of Blum and Mitchell's experiments with co-training on web page classification, we have applied this algorithm to email classification. We took 3 largest folders from one of the authors' inbox and formed 3 classification problems. For each problem one of the folders was considered to consist of interesting messages and others represented uninteresting email.

The only condition we have in the co-training settings is the presence of redundantly sufficient features that describe the data. Obviously, we could divide any email into two parts: a header and a body. In our experiments we used only a subject line instead of the whole header of a message because our 3 folders were formed mostly by the recipient's address, so the data are easily classified using only this information. Thus, our two sets of features are the words from the subject lines and the words from the bodies of email messages. For co-training to work these two sets of features have to be conditionally independent given the class. Generally, this condition is not always true for email domain. People often repeat the words from the subject in the body of a message. This violation is even stronger than it is in the web classification task because a subject and a body are written by the same person while a web page and hyperlinks pointed to it are usually created by different people.

For our experiments we had 1500 emails grouped into three folders with 250, 500, and 750 messages respectively. As was mentioned above, from these three groups we formed three 2-class problems (with positive and negative classes) by making positive examples from one group and negative from other two groups. Therefore, we had three classification problems with the distribution of positive and negative examples as 1:5 (highly imbalanced problem), 1:2 (moderately imbalanced problem), and 1:1

(balanced problem). The bigger imbalance in data, the worse learning results one might expect. It is due to the fact that most learning algorithms try to maximize the accuracy of predictions, which could be the highest for the trivial classifier that labels all examples with the majority class.

For each task 25% of the examples were left as a test set. From the remaining examples we randomly picked up p_0 positive and n_0 negative examples to form a training set (15 positive and 75 negative for the highly imbalanced problem, 5 positive and 10 negative for the moderately imbalanced problem, 3 positive and 3 negative for the balanced problem). Others were considered unlabeled.

The words from subjects and bodies were preprocessed; stop words were removed[4], the words that appear only in one message were removed, the remaining words were stemmed[5]. These preprocessed words form the feature sets. For each feature/word we count the number of times the word appears in the text to get the feature value.

In the experiments we ran 50 co-training iterations adding on each iteration one positive and five, two, or one negative examples for the highly imbalanced problem, the moderately imbalanced problem, and the balanced problem respectively (to maintain the initial distribution of positive and negative examples)[6]. Each experiment was conducted 10 times with different training/test splits. Each run started with at least default accuracy, which is 83.33% for the highly imbalanced problem, 66.67% for the moderately imbalanced problem, and 50% for the balanced problem. The results presented in tables are the averages of 10 runs. We used the implementations of learning algorithms Naive Bayes and Support Vector Machines from WEKA[20].

To get an upper bound for these experiments we trained Naive Bayes on a labeled version of the unlabeled data (see Table 1). As you can see from these results, the highly imbalanced problem appeared to be very hard for Naive Bayes as it cannot achieve on average the default accuracy (83.33%) even with all available labeled data.

Table 2 and Figure 1 show the results for the co-training experiment when the internal learning algorithm is Naive Bayes. The values in the table represent the absolute difference between the initial accuracy (the accuracy of a weak initial classifier trained on p_0 positive and n_0 negative examples) and the accuracy of the classifier that we get after 50 co-training iterations. The second column shows the differences for the subject-based classifier, the third column is for the body-based classifier, and the forth column is for the combined classifier. The combined classifier is obtained by picking the most confident predictions from the outcomes of the subject-based and the body-based classifiers. Almost all numbers in Table 2 are negative. It means that accuracy on average goes down, and co-training works in the opposite direction degrading the performance of the initial classifier. From the Figure 1 we can see that for the highly imbalanced problem adding new self-labeled examples hurts the performance badly on each iteration causing the learning curve go down quite rapidly. For the other two tasks we see a little increase (3-5%) in performance for the first 5-20 iterations, and then the learning curve goes down even below the initial accuracies.

One of the baselines for this experiment could be the co-training-like algorithm with random labeling. It differs from co-training only in that it adds not the most confident predictions but examples with random labels. This baseline is about -13% for the combined classifier for each task. So, co-training with Naive Bayes does better than this baseline for two out of three tasks. Another baseline could be the Expectation-Maximization (EM) algorithm[6] run on whole messages. Comparing with EM we could see whether co-training really benefits from having two redundantly sufficient sets of features in the email domain. The results for EM are shown in Table 3. Co-training does worse than EM on two out of

[4]Stop words are very frequent English words like pronouns, prepositions, etc. Lists of stop words are generally available (see, for example, http://www.superjournal.ac.uk/sj/application/demo/stopword.htm).

[5]Stemming is the process of removing suffixes and endings of words.

[6]We varied parameters of co-training such as the number of training examples and the number of examples added on each iteration but it does not influence the results significantly.

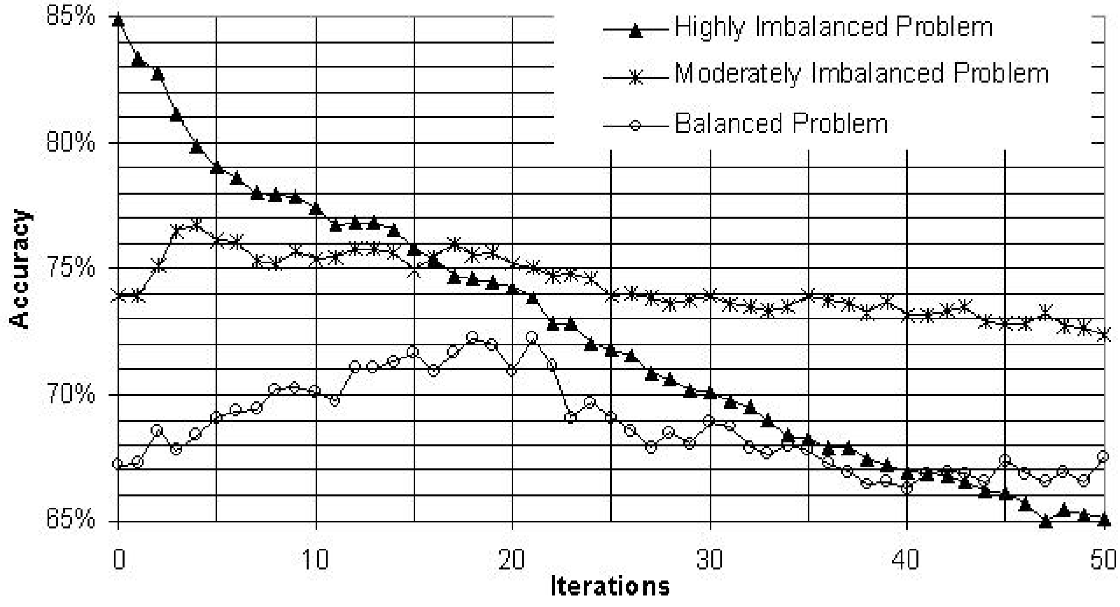

Figure 1: Co-Training with Naive Bayes

three tasks.

Why does Naive Bayes work so badly on our tasks? One of the possible reasons could be the violation of the conditional independence of the feature sets. To check if it is the real cause of such behavior, we removed the words that appear in the subject from the body of the messages and ran the experiment again. The results did not change much.

The other possible reason is that subjects and bodies are not redundant and needed to be used together for correct classification. For example, some users start their messages in a subject line and continue in the body or even leave the body empty if the message is short. Another example is that sometimes for quick access to the target email address people generate a new message as a reply for a previous message, change the body, but forget to change the subject. In that case the subject and the body could correspond to completely different topics.

The third possible reason is the great sparseness among the feature values. Most of the values are zeros because people tend to use very different words. To avoid zero probabilities we apply Laplace smoothing. However, the number of features is quite big. Naive Bayes may be quite sensitive to the number of features partly because of the violation of the independence assumption. Therefore, we tried to decrease the number of features by selecting a few the most important ones [7]. The results are presented in Table 4. We could see that with feature selection Naive Bayes works a lot better, and it benefits from co-training in two out of three tasks. The first task is the hardest because of the skewed distribution of examples but even this task looks better with feature selection (-2% compared with -20%).

The empirical test showed that the sparseness of the feature values is one of the reasons why co-training does not work with plain Naive Bayes. To confirm this and also to check if co-training can be applied in general to email classification, we changed the internal learning algorithm. Instead of Naive Bayes we have used Support Vector Machines (SVM)[18]. One of the characteristics of SVM important for us is that it works well even with very large feature

[7]We used correlation-based feature subset selection from WEKA[7].

	Naive Bayes	SVM
highly imbalanced problem(1:5)	80.36%	90.06%
moderately imbalanced problem(1:2)	83.98%	94.85%
balanced problem	91.51%	94.01%

Table 1: Naive Bayes and SVM trained on all available data

	Absolute difference in accuracy between the 1st and the 50th iterations		
	Subject-based classifier	Body-based classifier	Combined classifier
highly imbalanced problem(1:5)	-17.11%	-19.78%	-19.64%
moderately imbalanced problem(1:2)	-9.41%	-1.20%	-1.23%
balanced problem	0.78%	-0.53%	-0.64%

Table 2: Co-Training with Naive Bayes

	Absolute difference in accuracy between the 1st and the 50th iterations
highly imbalanced problem(1:5)	-6.64%
moderately imbalanced problem(1:2)	-1.71%
balanced problem	2.86%

Table 3: EM on whole messages

	Absolute difference in accuracy between the 1st and the 50th iterations		
	Subject-based classifier	Body-based classifier	Combined classifier
highly imbalanced problem(1:5)	-1.09%	-1.82%	-1.96%
moderately imbalanced problem(1:2)	1.62%	4.31%	3.89%
balanced problem	1.54%	6.78%	3.81%

Table 4: Co-Training with Naive Bayes and feature selection

	Absolute difference in accuracy between the 1st and the 50th iterations		
	Subject-based classifier	Body-based classifier	Combined classifier
highly imbalanced problem(1:5)	0.28%	1.80%	1.80%
moderately imbalanced problem(1:2)	14.89%	22.47%	17.42%
balanced problem	12.36%	15.84%	18.15%

Table 5: Co-Training with SVM

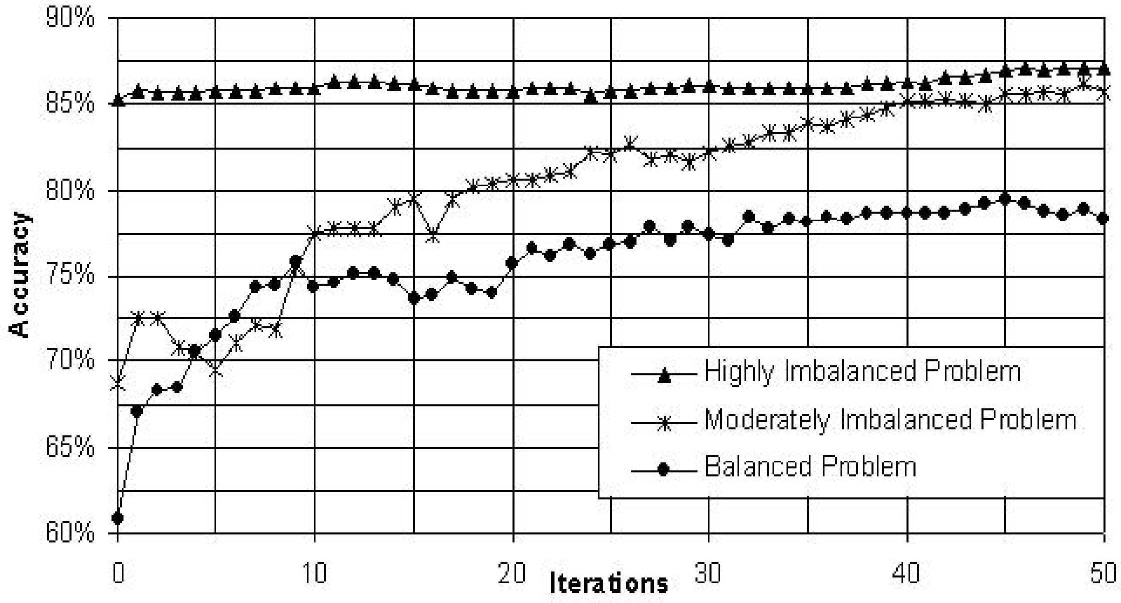

Figure 2: Co-Training with SVM

sets. Therefore, SVM is well-suited for text classification[8].

First, we tested SVM on a labeled version of the unlabeled data (see Table 1). In comparison with Naive Bayes, SVM did a lot better giving the evidence that this learning algorithm is a better choice for text classification (at least for our tasks).

Table 5 and Figure 2 present the results of co-training with SVM. All numbers there are positive, so even for the hardest task co-training with SVM improves the initial classifier. The learning curve for the highly imbalanced problem has a small but positive angle of slope. For the other two tasks, the curves go up quite sharply the first 10 iterations, and then their slopes become lower-grade. These results give us the evidence that co-training could be applied to email classification, and it could make the gain of up to 20% for some tasks.

5 Conclusion

In this research, we have presented a learning technique introduced in [1] which greatly decreases the effort needed in applying machine learning on real-life data. We have empirically proved that co-training can be applied to email classification. At the same time, we showed that the performance of co-training depends on the learning method it uses. Namely, Naive Bayes performed very poorly in our experiments while Support Vector Machines worked very well.

Though, more research is needed to clarify the causes of the poor behavior of Naive Bayes in combination with co-training and explore other possibilities (along with feature selection) to improve the performance of Naive Bayes in the co-training loop. For example, smoothing the probability distribution of Naive Bayes predictions could be helpful.

Another direction in this research is applying co-training to other domains. We plan to use the algorithm on grain classification. The task is to classify types of grains to predict the amount of harvest (big, medium, small) they could give. We have two different sources of information about the grains: their physical characteristics and their characteristics in infrared that could be transferred into two redundantly sufficient feature sets.

Also, a thorough comparative study of co-training and other learning methods that employ unlabeled data would be beneficial for the Machine Learning community.

Acknowledgements

This work has been supported by Communications and Information Technology Ontario (CITO) and AmikaNow! Corporation(www.amikanow.com).

About the Authors

Svetlana Kiritchenko is a graduate student in Computer Science at the University of Ottawa (Canada). She has a M.Sc. from Moscow State University (Russia) with major in Artificial Intelligence. Her research interests are in text classification and data mining.

Stan Matwin is a professor of Information Technology and Engineering and the director of the Graduate Certificate on Electronic Commerce program at the University of Ottawa. As well, he is the former head of the Canadian Society for Computational Studies of Intelligence, and IFIP WG 12.2 (Machine Learning).

Dr. Matwin's research interests are in data and text mining and knowledge-based systems. He has authored and co-authored some 100 research papers in refereed conferences and journals and is particularly interested in applied research which tackles practical problems in need of a solution.

References

[1] Avrim Blum and Tom Mitchell. Combining Labeled and Unlabeled Data with Co-Training. In *Proc. of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

[2] Gary Boone. Concept Features in Re:Agent, an Intelligent Email Agent. In *Proc. of the the 2nd International Conference on Autonomous Agents*, pages 141–148, St. Paul, MN, USA, 1998.

[3] Jake D. Brutlag and Christopher Meek. Challenges of the Email Domain for Text Classification. In *Proc. of the 17th International Conference on Machine Learning*, pages 103–110, Stanford University, USA, 2000.

[4] William W. Cohen. Learning Rules that Classify Email. In *Proc. of the AAAI Spring Simposium on Machine Learning in Information Access*, 1996.

[5] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to Construct Knowledge Bases from the World Wide Web. *Artificial Intelligence*, (118):69–113, 2000.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[7] M. A. Hall. *Correlation-based Feature Subset Selection for Machine Learning.* PhD thesis, University of Waikato, 1998.

[8] Thorsten Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proc. of the 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, Germany, 1998.

[9] Thorsten Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *Proc. of the 16th International Conference on Machine Learning*, pages 200–209, San Francisco, USA, 1999.

[10] Thorsten Joachims. *The Maximum Margin Approach to Learning Text Classifiers: Methods, Theory, and Algorithms.* PhD thesis, Universität Dortmund, 2000.

[11] George H. John and Pat Langley. Estimating Continuous Distributions in Bayesian Classifiers. In *Proc. of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 338–345, Montreal, Quebec, Canada, 1995. Morgan Kaufmann.

[12] Ion Muslea, Steven Minton, and Craig A. Knoblock. Selective Sampling + Semi-Supervised Learning = Robust Multi-View Learning. In *IJCAI-2001 Workshop "Text Learning: Beyond Supervision"*, 2001.

[13] Kamal Nigam and Rayid Ghani. Analyzing the Effectiveness and Applicability of Co-training. In *Proc. of the 9th International Conference on Information Knowledge Management*, pages 86–93, McLean, VA , USA, 2000.

[14] David Pierce and Claire Cardie. Limitations of Co-Training for Natural Language Learning from Large Datasets. In *Proc. of the 2001 Conference on Empirical Methods in Natural Language Processing*, CMU, Pittsburgh, PA, USA, 2001.

[15] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian Approach to Filtering Junk E-mail. In *AAAI-98 Workshop on Learning for Text Categorization*, Madison, Wisconsin, USA, 1998.

[16] Richard B. Segal and Jeffrey O. Kephart. MailCat: An Intelligent Assistant for Organizing E-Mail. In *Proc. of the Sixteenth National Conference on Artificial Intelligence*, pages 925–926, Orlando, Florida, USA, 1999.

[17] L. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[18] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[19] Werner Winiwarter. PEA - a Personal Email Assistant with Evolutionary Adaptation. *International Journal of Information Technology*, 5(1), 1999.

[20] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999. http://www.cs.waikato.ac.nz/ ml/weka/.

[21] Sarah Zelikovitz and Haym Hirsh. Improving Short-Text Classification using Unlabeled Background Knowledge to Assess Document Similarity. In *Proc. of the 17th International Conference on Machine Learning*, Stanford University, USA, 2000.