Fast Uncertainty Sampling for Labeling Large E-mail Corpora

Richard Segal
IBM Research
Hawthorne, New York, USA
rsegal@us.ibm.com

Ted Markowitz
Pace University
White Plains, New York, USA
tjm@cognosys.net

William Arnold
IBM Research
Hawthorne, New York, USA
barnold@us.ibm.com

ABSTRACT

One of the biggest challenges in building effective anti-spam solutions is designing systems to defend against the ever-evolving bag of tricks spammers use to defeat them. Because of this, spam filters that work well today may not work well tomorrow. The adversarial nature of the spam problem makes large, up-to-date, and diverse e-mail corpora critical for the development and evaluation of new anti-spam filtering technologies. Gathering large collections of messages can actually be quite easy, especially in the context of a large, corporate or ISP environment. The challenge is not necessarily in collecting enough mail, however, but in collecting a representative distribution of mail types as seen "in the wild" and in then accurately labeling the hundreds of thousands or millions of accumulated messages as spam or non-spam. In the field of machine learning *Uncertainty Sampling* is a well-known *Active Learning* algorithm which uses a collaborative model to minimize the human effort required to label large datasets. While conventional Uncertainty Sampling has been shown to be very effective, it is also computationally very expensive since the learner must reclassify all the unlabeled instances during each learning iteration. We propose a new algorithm, *Approximate Uncertainty Sampling* (AUS), which is nearly as efficacious as Uncertainty Sampling, but has substantially lower computational complexity. The reduced computational cost allows Approximate Uncertainty Sampling to be applied to labeling larger datasets and also makes it possible to update the learned model more frequently. Approximate Uncertainty Sampling encourages the building of larger, more topical, and more realistic example e-mail corpora for evaluating new anti-spam filters. While we focus on the binary labeling of large volumes of e-mail messages, as with Uncertainty Sampling, Approximate Uncertainty Sampling can be used with a wide range of underlying classification algorithms for a variety of categorization tasks.

1. INTRODUCTION

To reliably test the efficacy and efficiency of e-mail classification algorithms, both researchers and anti-spam application developers require access to large pools of recent e-mail messages. Existing, publicly-available corpora are mostly small (in the thousands or ten's of thousands of sample messages), many are already several years old and continue to lose more and more of their relevance with the passage of time, and finally, a number of them contain either nearly all spam e-mail or have an insufficient number of the good mail examples required to accurately measure performance at low false positive rates. Moreover, these public corpora often have inherent sampling biases that make them unrepresentative of the e-mail traffic which is seen in real world environments.

The creation of labeled e-mail corpora for effectively testing anti-spam applications can be a difficult undertaking. The data used to build new corpora should be selected to match as closely as possible a filter's intended audience and care must be taken not to introduce any biases into the sampling or labeling process that might affect the eventual evaluation results. Optimally, test corpora should be created by randomly sampling representative messages from an appropriate data source and accurately labeling each and every message as either spam or non-spam. However, labeling messages by visual inspection to build even a small corpus can be a tedious, error-prone, and expensive proposition. As a result, the hand-labeling of large numbers of messages—hundreds of thousands or millions—to produce an accurate corpus is virtually impossible under normal circumstances.

In practice, most corpora are created using one of three mechanisms. The first is to enlist several users to manually label all their spam for some specified length of time. Any message not labeled as spam is assumed to be good mail. It is critical that the user does not use any technology to help separate spam from good mail such as a "Junk Folder" as this can introduce bias into the labeling process. But when implemented correctly, it can be very effective for building corpora for evaluating personalized spam filters. The second method is to rely on user-initiated feedback such as asking the user to express their opinions via special "this is spam" or "this is not spam" voting buttons in their e-mail client. The downside of this approach is that users often have marked tendencies as to which messages they actually label, such as focusing only the most egregious spam or rarely, if ever, bothering to proactively label good mail. A third technique is the use of honeypots or spamtraps to automatically capture examples of spam, however these tools are usually limited to identifying only spam e-mail and do not provide the balanced corpus of both spam and non-spam examples needed to evaluate most algorithms.[1]

In the best of all possible worlds the content of e-mail test corpora and their creation process should have the following characteristics:

- Be as accurately labeled as possible
- Be as large as is practicable to more effectively measure the small false positive rates required by spam filters

[1]The use of user-initiated feedback can be very effective for providing additional training examples for a live anti-spam system. The issues raised here apply only to the use of user feedback to create test corpora for evaluation purposes.

- Be diverse with regard to message size, format, content, MIME structure and header contents, representing a wide range of message types

- Be current with changing message features and evolving spammer techniques

- Be consistent with the end-user's actual e-mail traffic to better reflect actual operating conditions

- Be representative of the target e-mail environments in the distribution of both spam *and* non-spam messages

- Be cheap to build—cheap, that is, in overall time spent, computational resources expended, and total amount of human adjudication effort required

Over the past several years our own anti-spam research group has addressed the need for a large testing corpus by meticulously labeling a database of approximately 180,000 corporate e-mail examples over a period of several months. This was done using a combination of ad-hoc methods, including the use of volunteers as the final arbiters of a message's classification when that became necessary. While this dataset has proven to be quite useful up to this point, the examples have naturally grown stale over time. As a result there is a constant need to be able to easily create more current and more varied message datasets for future experiments. Our goal is to create a new corpus of at least 1,000,000 messages. Fortunately, obtaining large volumes of e-mail is not an issue in this environment. In fact, our archive currently contains over 20 million messages just waiting to be labeled with more becoming available daily. The overarching impetus behind the work described in this paper is the desire to virtually eliminate the need for human intervention in the labeling process and to automate the building of current, fresh corpora.

2. ACTIVE LEARNING APPROACH

As a starting point for this corpus-building effort we've adopted a common machine learning technique, *active learning*.[1] Active learning is a form of inductive machine learning whereby the learning model proactively queries a trusted party—often referred to as the "teacher" or "oracle"—for examples of potential interest, that is, examples which the learner *itself* thinks might strengthen its future ability to make more accurate predictions on as yet unseen examples. The defining characteristic of active learning is that, while there is some give-and-take between the knowledgeable teacher and the questioning learner, it is primarily learner-driven. In other words, the learning model incrementally improves its discrimination ability by constantly re-evaluating its own performance—using some internal effectiveness metrics, e.g., the information gain/loss contributed by each example—based on what questions it has asked to that point. Only after this introspection step does the learner select what it thinks are the next set of "best" examples to ask for during the next iteration.

Figure-1 illustrates the basic active learning algorithm applied to corpus labeling. The labeler has complete access to the contents of a pool of unlabeled messages U during each learning iteration as well as the ability to ask about another batch of examples of sample size M. In this domain L represents a set of e-mail messages already labeled by the teacher. Based on the answers (*"spam"* vs. *"good"*) the labeler receives from its questions (*"What was the label for Message-ID=X?"*), the labeler reassesses what it's learned thus far and adjusts its next batch of questions accordingly. The goal of this process is to shrink the amount of human intervention involved by minimizing the total number of questions asked

```
FUNCTION ActiveLearningLabeler(U, M, C) returns L
    // Input
    //     U: Set of unlabeled data
    //     M: Batch size of query set
    //     C: Untrained classifier to use to label data
    // Output
    //     L: Labeled data
    While not done do
        Q = SelectQuery(C, U, M)
        Foreach q in Q do
            l = AskTeacher(q)
            U = U - q
            L = L ∪ (q, l)
            TrainClassifier(C, q, l)
        EndFor
    EndWhile
    Foreach e in U do
        l = Classify(C, e)
        L = L ∪ (e, l)
    EndFor
    Return L
END
```

Figure 1: Active-learning algorithm for labeling large corpora

by the labeler, while simultaneously increasing the labeler's predictive power over time in terms of the total number of correctly labeled messages, fewest false negatives (spams incorrectly labeled as good) and fewest false positives (good email incorrectly labeled as spam).

3. UNCERTAINTY SAMPLING

Active learning as the basis for labeling unlabeled data has been implemented in a number of ways using a range of classification techniques. *Uncertainty Sampling* is one form of active learning that has been used extensively for labeling unlabeled data since it was first proposed in the mid-1990's. This technique, as described by Lewis and Gale [8], utilizes a single classifier to generate a probability value $P(C|w)$ for an example, where C is the set of classification categories and w is a feature vector representing an observed pattern. This probability is called the example's *certainty score*. The score is a continuous value from 0.0 to 1.0 with 0.5 denoting complete uncertainty vis a vis what the category should be. The intuition here is that choosing the next sample based on the questions that the labeler was most unsure of will bias the labeler toward faster learning. For more theoretical background on the effectiveness of active learning, see [13].

Uncertainty Sampling has shown itself to be quite effective, but it can also incur high computational costs, especially when run on large volumes of unlabeled data. What follows is a paraphrase of the formal algorithm depicted in Figure-2:

While the teacher is willing to label examples...

1. Apply the current labeler model to each unlabeled example

2. Find the M examples for which the labeler is least certain of class membership

3. Have the teacher label that M-sized set of sample questions

4. Train a new labeler on all labeled examples

```
FUNCTION SelectQuery_US(C, U, M) returns Q
    //  Input
    //      C: Classifier trained on previously labeled data
    //      U: Unlabeled data
    //      M: Batch size of query set
    //  Output
    //      Q: Set of examples to label
    Foreach e in U do
        Let R(e) = ComputeUncertainty(C, e)
    EndFor
    Let Q = Select the M examples from U that maximize R().
    Return Q
END
```

Figure 2: Uncertainty Sampling

Because the cost for rerunning the learner on all the data each time in step 2 can become prohibitive, the algorithm is often implemented to use large batches of questions during each iteration. Using these large batches can make the algorithm's learning rate more sluggish however, since it cannot then respond as quickly to new data provided by the teacher.

From the point of view of computational complexity the original Uncertainty Sampling algorithm runs in $O(I \cdot N)$, where $I = \frac{W}{M}$ (W is the total number of queries required of the teacher to meet a specific error rate and M is the batch size) and N represents the total size of the corpus being labeled. In the next section we will suggest an improvement on Uncertainty Sampling that, while it does not reach the same labeling accuracy levels quite as fast as the original algorithm, can make the labeling of much larger corpora with much less human attention more feasible.

4. APPROX. UNCERTAINTY SAMPLING

Uncertainty Sampling is computationally prohibitive because at each iteration it must evaluate the uncertainty of every unlabeled message on the current classifier. Random sampling is not a good alternative as it converges too slowly for a human to label large e-mail corpora. We propose Approximate Uncertainty Sampling as a compromise that converges almost as fast as true Uncertainty Sampling, but requires substantially less computational resources.

Uncertainty Sampling re-evaluates the uncertainty of each message during each iteration. However, the uncertainty of most messages will not change significantly as the result of new training data. This is particularly true of messages where the labeler already has a high certainty in its current classification. One method to reduce the computational cost of Uncertainty Sampling is to re-evaluate only a subset of the unlabeled messages during each iteration. A second opportunity for improving the efficiency of Uncertainty Sampling is to note that it works very hard each cycle to select the best m queries when a near-best set of queries will often do almost as well.

At each iteration, Approximate Uncertainty Sampling selects only a subset of examples to re-evaluate and then chooses the best m examples amongst this limited subset. The key to the effectiveness of Approximate Uncertainty Sampling is how it selects the subset of examples to consider in each iteration. Approximate Uncertainty Sampling applies the Uncertainty Sampling concept recursively so that the subset of examples considered is determined by applying Uncertainty Sampling to the existing subset of unlabeled messages. But rather than re-evaluate the uncertainty of each message, the sampling is performed using the uncertainties calculated in previous iterations.

```
FUNCTION SelectQuery_AUS(C, U, M, R) returns Q, R
    //  Input
    //      C: Classifier trained on previously labeled data
    //      U: Unlabeled data
    //      M: Batch size of queries
    //      R: Previously recorded uncertainty values
    //  Output
    //      Q: Set of examples to label
    //      R: Updated uncertainty values
    Let T = M*log(|U|)
    Let S = Select the T examples from U that maximize R().
    Foreach e in S do
        Let R(e) = ComputeUncertainty(C, e)
    EndFor
    Let Q = Select the M examples from U that maximize R().
    Return Q
END
```

Figure 3: Approximate Uncertainty Sampling

Figure-3 presents the Approximate Uncertainty Sampling algorithm in detail. In addition to the active classifier, unlabeled data, and batch size, Approximate Uncertainty Sampling takes a fourth parameter R that stores the last recorded uncertainty value for each unlabeled message. When it is first called, R is initialized by setting all examples to have maximal uncertainty.

Approximate Uncertainty Sampling begins by selecting $\mathtt{S} \subset \mathtt{U}$, the $M \log N$ most uncertain examples based on the uncertainties stored in R. The uncertainty of each example in S is then re-evaluated using the current, most up-to-date classifier. The resulting uncertainty values are stored in R for use in future iterations. Finally, the M most uncertain examples are selected from S to form the set of queries that are returned to the user.

The key to both the effectiveness and efficiency of Approximate Uncertainty Sampling is the size of the subsample chosen during each iteration. On the one hand, the larger the sample size, the closer the algorithm approximates full Uncertainty Sampling. And on the other hand, the smaller the sample size chosen, the more efficient the algorithm.

To balance these, Approximate Uncertainty Sampling uses a sample size of $M \log N$ as a good compromise between optimal query selection and computational complexity. The computational complexity of Approximate Uncertainty Sampling is:

$$O\left(I \cdot M \log N\right) = O\left(\frac{W}{M} \cdot M \log N\right) = O\left(W \cdot \log N\right)$$

As its computational complexity grows only logarithmically in N, Approximate Uncertainty Sampling can scale to labeling much larger datasets than those that are practical with full Uncertainty Sampling which is $O(N)$. In addition, its computational complexity for labeling W examples is independent of the batch size M. This makes Approximate Uncertainty Sampling suitable for using very small batch sizes and even for re-sampling after every message labeled by the user.

While a subsample size of $M \log N$ does provide a good balance between labeling efficiency and computational complexity, the basic ideas of Approximate Uncertainty Sampling can be applied to any size subsample. In an interactive labeling scenario where a human is labeling each query set, Approximate Uncertainty Sampling can be easily adapted to function as an anytime algorithm that continuously improves its next query set until the human has finished labeling the previous set of examples.

5. EXPERIMENTAL EVALUATION

Unfortunately, we cannot evaluate these labeling algorithms directly against our target corpus of 1,000,000 messages because there is as yet no easy way to absolutely validate the accuracy of all the labels without personal inspection. Instead, we evaluate the effectiveness of the algorithms by running our experiments using an existing public dataset, the TREC 2005 Spam Track Corpus [2], for which a "gold standard" labeling already exists. This dataset consists of over 92,000 messages, of which about 60% are labeled spam and 40% are labeled non-spam.

We use this labeled corpus to measure the effectiveness of each algorithm in a simulated active learning framework that mimics how we expect these algorithms to be used in practice. A priori the labeler is given the contents of the full TREC dataset, but with all the message labels removed. At each step of the simulation, the labeler is asked for the set of queries that should be labeled next by the teacher. The simulator—standing in for the human teacher in this scenario—then marks the messages according to the known gold standard labels and passes the labels back to the labeler so that it can update its current classifier model. At fixed intervals, the simulation is interrupted to evaluate the performance of the labeler. The evaluation is accomplished by asking the labeler to label the entire dataset and then recording the accuracy of that labeling with respect to the known gold standard. During the evaluation, the labeler is not informed as to the correct label of any examples.

In order to reduce any potential bias created by the input dataset's ordering or distribution of spam vs. good e-mail examples, we randomized the list of examples in the original TREC corpus and then split that list into five separate subsets. Each subset contains 80% ($\approx$74K) of the total 92K examples, with every one having a different 20% of the total original examples removed. The results shown below are based on the average performance across all five subsets of the examples.

All the test labelers were given the same classifier to build their learning models. The classifier used is a boosted Naïve Bayes classifier, but modified to use geometric mean rather than arithmetic mean to combine the conditional probabilities of each word. This classifier was chosen because it is highly competitive with existing solutions and because its particular implementation is quite fast and efficient, allowing us to run more experiments on larger datasets. Other types of text classification algorithms, e.g., decision-trees [3], Support Vector Machines [6], or Expectation-Maximization (EM) techniques [9], could be used just as well with Approximate Uncertainty Sampling for this task.

Figure 4 shows the results of applying this methodology to evaluate several active learning algorithms on the TREC 2005 public corpus. A batch size of $M = 100$ was used for this experiment to make the computational costs of running the experiment manageable. The results show that both Approximate Uncertainty Sampling and full Uncertainty Sampling perform very well on this dataset. Both achieve error rates below 0.25% within less than 3,000 queries. For comparison, random sampling does not achieve this level of accuracy even after 15,000 queries. While the performance of Approximate Uncertainty Sampling is good, it does not quite match the strong performance of Uncertainty Sampling. The error rate of Approximate Uncertainty Sampling is about twice that of true Uncertainty Sampling over the range of 1,000 to 5,000 queries. After 5,000 queries however, both algorithms converge to nearly identical error rates. Approximate Uncertainty Sampling requires about twice the number of queries as does Uncertainty Sampling to achieve a given error rate. These results demonstrate that

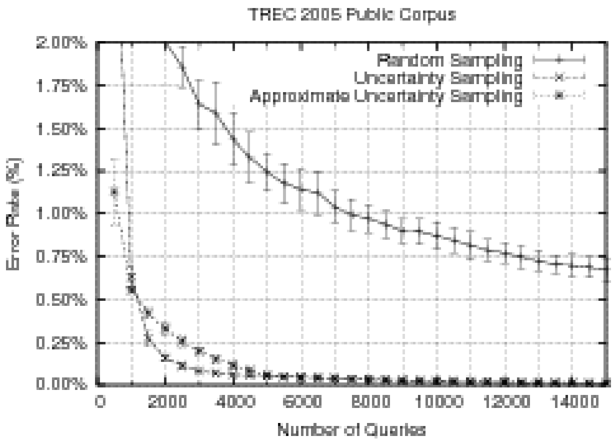

Figure 4: Effectiveness of Active Learning

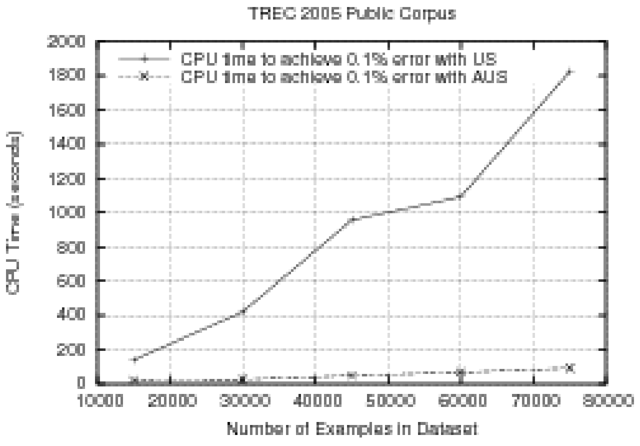

Figure 5: CPU Time to Achieve 0.1% Error Rate

Uncertainty Sampling is still the better algorithm when it is computationally feasible.

As seen in Figure 5 using our highly optimized, in-memory classifier implementation, Approximate Uncertainty Sampling averaged less than 2 minutes of processing time to achieve a 0.1% error rate. In comparison, Uncertainty Sampling required over 30 minutes of processing to match the same error rate using the exact same classifier configuration.

Figure 6 shows the number of queries required to reach a given error rate using Approximate Uncertainty Sampling for various dataset sizes. The experiments suggest a roughly linear relationship between the size of the dataset and the number of queries. Extrapolating out from the 4,500 queries that it takes to achieve a 0.1% error rate on 74,000 examples, we find that the number of queries needed to achieve a similar error rate on 1,000,000 examples would be near 60,000 queries. Likewise, Figure 7 shows that the number of queries required by Uncertainty Sampling also grows linearly with dataset size. Using the same target of 1,000,000 examples Uncertainty Sampling would require approximately 40,000 queries. Unfortunately this is beyond what a small team could reasonably hand-label even with a week or two of devoted effort. Given these data points it appears that neither algorithm is truly up to the task of labeling a 1,000,000 message corpus as-is.

Further improvements are obviously required to make labeling

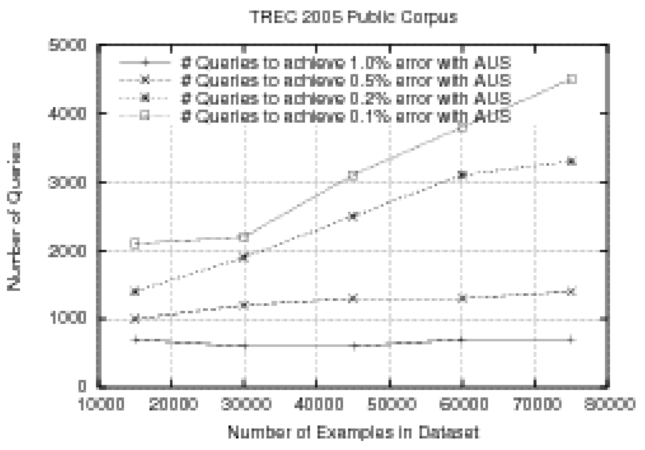

Figure 6: Approximate Uncertainty Sampling Effectiveness

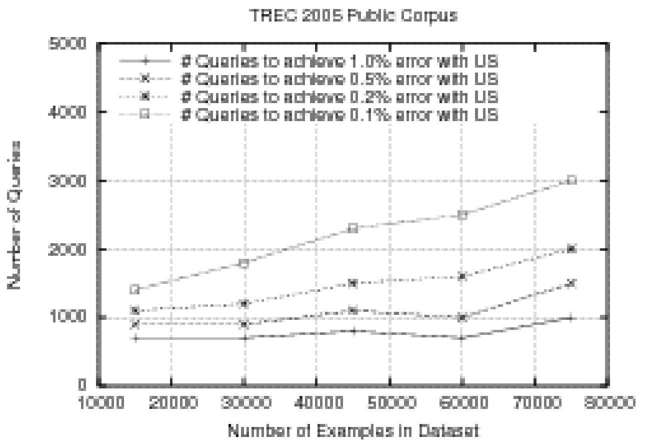

Figure 7: Uncertainty Sampling Effectiveness

these large corpora tractable. However, Approximate Uncertainty Sampling, because of its computational efficiency vis-à-vis pure Uncertainty Sampling, creates an opportunity for applying other CPU-intensive techniques to augment the labeling process, such as, Error Reduction Sampling [10], the use of multiple classifiers, classifier aggregation, and/or clustering.

6. RELATED WORK

The benchmark algorithm for labeling unlabeled data is Uncertainty Sampling [7, 8]. Since its introduction there have been many enhancements to the basic algorithm. These include Weighted Uncertainty Sampling [11], Error Reduction Sampling [10], and Adaptive Sampling [5]. Most of these perform additional computation to choose a better sample in order to reduce the number of queries required to achieve a given error rate. The algorithm described in this paper could be adapted to improve the computational complexity of many of these other algorithms.

Similarly, other labeling techniques, e.g., *Query By Committee* (QBC) [12], use combinations of multiple learners to decrease the volume of questions required during labeling. We also see an opportunity for synergies with these types of labelers.

Hulten *et. al.* [4] describe an alternative technique for creating large, unbiased corpora by making use of the collaborative efforts of one hundred-thousand volunteer MSN Hotmail users. Every day, a random message is selected from the mail stream of each volunteer and the volunteer is asked to label the message as spam or not spam. The Hotmail Feedback Loop produces tens of thousands of labeled messages each day, which would completely label our initial target of one million messages in about three months. The main limitation of this approach is that it is only viable for large ISP's or other very large organizations where creating a network of one hundred-thousand or more volunteers is feasible.

The importance and the challenges involved in building a large, labeled spam-related corpus is illustrated by the development of the TREC 2005 Spam Track dataset [2]. The TREC corpus — created using publicly-available Enron e-mails — was completed after eight phases of refinement including the use of several different classification algorithms, as well as hand-adjudication. The goal of our work is to make this type of corpus-building easier.

7. CONCLUSION

Building large, accurately labeled e-mail corpora for training and testing anti-spam tools is hard. While accumulating large volumes of e-mail is fairly straightforward if one has ready access to e-mail messages, we cannot expect their correct categorization to be left to human hands alone. Machine learning-based tools can be used to reduce the total amount of time spent in human adjudication of unlabeled messages. An appropriate balance must be struck between the computational costs of these machine learning algorithms and the burden placed on the human labelers. We offer a variation on a well-known algorithm with lower computational complexity which can substantially reduce these processing costs. Also, because it is an anytime algorithm it can asynchronously continue to process data, improving on and minimizing the set of queries eventually delivered to the human labeler.

8. REFERENCES

[1] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press, 1995.

[2] G. Cormack and T. Lynam. Spam corpus creation for trec. In *Second Conference on Email and Anti-Spam CEAS 2005, Stanford University, Palo Alto, CA*, 2005.

[3] S. Goldman and Y. Zhou. Enhancing supervised learning with unlabeled data. *Proceedings of the Seventeenth International Conference on Machine Learning: ICML-2000*, 2000.

[4] G. Hulten, J. Goodman, and R. Rounthwaite. Filtering spam e-mail on a global scale. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 366–367, New York, NY, USA, 2004. ACM Press.

[5] V. Iyengar, C. Apte, and T. Zhang. Active learning using adaptive resampling. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 91–98, 2000.

[6] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, 1998.

[7] D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In W. W. Cohen and H. Hirsh, editors, *Proceedings of ICML-94, 11th International Conference on Machine Learning*, pages 148–156, New Brunswick, US, 1994. Morgan Kaufmann Publishers, San Francisco, US.

[8] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In W. B. Croft and C. J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.

[9] K. Nigam. *Using Unlabeled Data to Improve Text Classification*. PhD thesis, Carnegie-Mellon University, Pittsburgh, US, 2001.

[10] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, San Francisco, CA, 2001.

[11] M. Saar-Tsechansky and F. Provost. Active sampling for class probability estimation and ranking, 2004.

[12] H. Seung, M. Opper, and H. Sompolinsky. Query by Committee. *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[13] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.