Improving Short-Text Classification Using Unlabeled Background Knowledge to Assess Document Similarity

Sarah Zelikovitz ZELIKOVI@CS.RUTGERS.EDU
Haym Hirsh HIRSH@CS.RUTGERS.EDU
Computer Science Department, Rutgers University, 110 Frelinghuysen Road, Piscataway, NJ 08854-8019 USA

Abstract

We describe a method for improving the classification of short text strings using a combination of labeled training data plus a secondary corpus of unlabeled but related longer documents. We show that such unlabeled background knowledge can greatly decrease error rates, particularly if the number of examples or the size of the strings in the training set is small. This is particularly useful when labeling text is a labor-intensive job and when there is a large amount of information available about a particular problem on the World Wide Web. Our approach views the task as one of information integration using WHIRL, a tool that combines database functionalities with techniques from the information-retrieval literature.

1. Introduction

The task of classifying textual data that has been culled from sites on the World Wide Web is both difficult and intensively studied (Cohen & Hirsh, 1998; Joachims, 1998; Nigam et al., 1999). Applications of various machine learning techniques that attempt to solve this problem include categorization of Web pages into sub-categories for search engines, and classification of news articles by subject. Machine learning programs, such as C4.5 (Quinlan, 1993) and RIPPER (Cohen, 1995) have the limitation that they learn based solely upon previously classified data. It is often both impractical and extremely tedious and expensive to hand-label a sufficient number of training examples to achieve the high accuracy that is needed for a given task. Given the huge proliferation of data on the Web, only a tiny percentage of which can realistically be classified and labeled, these programs are unable to exploit this information to achieve higher accuracy when faced with new unlabeled examples.

Various researchers in text learning and mining have recognized that although there might be very few labeled examples, there can be a tremendous amount of unlabeled examples. Nigam et al. (in press) have done work on this using Expectation Maximization (EM) and a naive Bayes classifier. The parameters of the naive Bayes classifier are set using labeled examples. The learned model is then used by EM to probabilistically classify unlabeled documents, with the resulting collection of classified documents used to estimate a new set of parameters for naive Bayes. The EM algorithm iterates until there is no change in the naive Bayes parameters. Nigam et al. present a number of experimental results that show that error rates can be reduced significantly using unlabeled examples in this way. Other related algorithms are described by McCallum and Nigam (1999) and Jones et al. (1999).

Blum and Mitchell's (1998) co-training algorithm also uses unlabeled data to improve learning. Their algorithm applies to problems where the target concept can be described in two redundantly sufficient ways (such as through two different subsets of attributes describing each example). Each view of the data is used to create a predictor, and each predictor is used to classify unlabeled data which are then used to further train the other learner. Blum and Mitchell prove that under certain conditions, the use of unlabeled examples in this way is sufficient to PAC-learn a concept given only an initial weak learner. Lewis and his colleagues (Lewis & Gale, 1994; Lewis & Catlett, 1994) also make use of unlabeled data in learning, but focus on asking for labels from a human labeler for only a modest subset of the data, those whose class membership is most undecided given the result of learning on the data that has been labeled thus far.

This paper also describes a method that uses a corpus of unlabeled data to assist in the classification task. However, unlike the preceding approaches, we do not attempt to classify the data, and, indeed, do not even require that it be of a form comparable to that of the training data. In many cases it is not even possible to classify the data using the classification schema of the labeled instances. Instead, we use the unlabeled corpus as "background knowledge" for the

learner, to aid it in its decision task. Rather than directly comparing a new unlabeled example directly to elements of the labeled training corpus, we use the unlabeled background knowledge as a "bridge", to connect the new example with labeled examples. A labeled training example is useful in classifying an unknown test instance if there exists some set of unlabeled background knowledge that is similar to both the test example and the training example. We call this a "second-order" approach to classification, in that data are no longer directly compared but, rather, are compared one step removed, through an intermediary.

In more detail, in this paper we look at improving the classification of short text strings by using unlabeled but related longer documents. A concrete example of the usefulness of our approach can be seen in the task of assigning topic labels to technical papers. In labeling the title of a physics article with its sub-specialty, any title containing a word such as *galaxy* should easily be classified correctly as an astrophysics paper, even if there are few training articles in that domain. However, an article on a less common topic, as for example *old white dwarfs*, would only be classified correctly if a title with these words appears in the labeled training examples. Although the training set does not contain the words *old white dwarf* in our experimental data, our system is able to correctly classify a title with these words as astrophysics, by utilizing a corpus of unlabeled paper abstracts from the same field, which is naturally available on the Web. In our "second-order" approach, our system finds those unlabeled paper abstracts that are most similar to both *old white dwarfs* and to various training titles. These training titles are then used to classify *old white dwarfs* correctly, although each of these titles is quite dissimilar to it when compared directly.

In order to achieve our goal, we use WHIRL (Cohen, 1998a; Cohen, 1998b) which is a conventional database system augmented with special operators for text comparison. Its use as a text classification program is a nearest neighbor approach (Cohen & Hirsh, 1998), with text documents specified as TFIDF vectors, and similarity between text documents measured as cosine similarity (Salton 1989). WHIRL makes it possible to pose SQL-like queries on databases with text-valued fields. If we consider the training examples as a table, the background knowledge as a table, and a test example as a table as well, WHIRL provides a framework in which we can easily specify and explore "second-order" similarity classification. It allows for succinct queries that specify the combination of training similarity and background similarity to a new test example.

In the next section we give a brief review on WHIRL and a discussion on how we use it for classification with unlabeled data. We then describe four distinctly different domains on which we tested our system. The domain descriptions are followed by a set of results for each of the domains for varied data sets. We conclude with a discussion of the various possible dimensions that our choices along the way can take and directions for current and future research.

2. Our Approach

2.1 WHIRL for Text Classification

WHIRL (Cohen 1998a; Cohen 1998b) is an information integration tool that is specifically designed to query and integrate varied textual sources from the Web. WHIRL's SQL-type queries can search and retrieve textual sources based upon specified conditions. Assume that we have a corpus of training examples with labels, and a test example that must be assigned a label. The training examples can be viewed as a table with the field *instance*, to hold the textual data, and field *label* to hold the class label. The test example is a one line table, with simply the textual field *instance*. An example of a WHIRL query (Cohen & Hirsh, 1998) is:

```
SELECT Test.instance, Train.label
  FROM Train AND Test
 WHERE Train.instance SIM Test.instance
```

Given a user-specified parameter *K*, this query will first generate an intermediate table containing the *K* tuples

$$\langle \textit{Test.instance}, \textit{Train.instance}, \textit{Train.label} \rangle$$

that maximize the similarity score between *Test.instance* and *Train.instance*. Unlike traditional SQL queries, the result of this is a set of tuples ordered by score, with the highest score representing the closest *Train.instance*, *Test.instance* pair, using WHIRL's SIM operator to compute the similarity of these textual documents.

To compute similarity WHIRL computes a model of each text document by representing each document as a vector in a vector space. This representation is computed by passing each document through a stemmer (Porter, 1980) and by then computing weights for each term using the TFIDF (Salton, 1989) weighting method. Distances between vectors are computed using the cosine metric, which represents the statistical similarity between documents.

In WHIRL's final step it takes this table of K tuples and projects it onto the fields specified in the SELECT statement. Note that this can mean that there may be many training examples among the K with the same label (i.e., multiple nearby examples in the training set), and these are combined into a single tuple in the final result table. The combination of scores is performed by treating scores as probability and the combination as a "noisy or." If the individual scores of the tuples with a given label are $\{s_1, \ldots, s_n\}$, the final score for that label is $1 - \prod_{i=1}^{n}(1 - s_i)$. Whichever label has the highest score in the resulting projected table is

returned as the label for the test instance. This method bears many similarities to the nearest-neighbor method of Yang and Chute (1994), which has been shown to perform quite well on text classification tasks (Cohen & Hirsh, 1998). Indeed, based on these two papers we also use a value of $K = 30$ in our experiments.

2.2 WHIRL with Background Knowledge

The question we now ask is: How can we use a large body of unlabeled data, or "background knowledge", to aid classification? Although these pieces of information need not be labeled, and indeed may have no relationship whatsoever to the classification task, we would hope to learn something from the word combinations in these examples. Such background knowledge may provide us with a corpus of text that contains information both about importance of words (in terms of their TFIDF values in this large corpus), and joint probability of words (what percentage of the time do two words coexist in a document?). This gives us a large context in which to test the similarity of a training example with a new test example. We can use this context in conjunction with the training examples to label a new example.

Because of WHIRL's expressive language, and the ability to create conjunctive queries simply by adding conditions to a query, WHIRL's queries for text classification can be expanded to allow for the use of "background knowledge" on a subject. In the example of the classification of physics paper titles discussed earlier, suppose that we had a fairly small set of labeled paper titles, and also a very large set of unlabeled titles, papers or abstracts (or Web pages resulting from a search), in a relation called Background with a single field, *value*. We can create the following query for classification

```
SELECT Test.instance, Train.label
  FROM Train AND Test AND Background
WHERE Train.instance SIM Background.value
  AND Test.instance SIM Background.value
```

Here each of the two similarity comparisons in the query computes a score, and WHIRL multiplies them together to obtain a final score for each tuple in the intermediate-results table. This table is then projected onto the *Test.instance* and *Train.label* fields as discussed before. Whichever label gives the highest score is returned as the label for the test example.

One way of thinking about this is that rather than trying to connect a test example directly with each training example, it instead tries to bridge them through the use of an element of the background table. Note that WHIRL combines the scores of tuples generated from different matches to the background table. Our use of WHIRL in this fashion thus essentially conducts a search for a set of items in the background knowledge that are close neighbors of the test example, provided that there exists a training example that is a neighbor of the background knowledge as well. Training neighbors of a test example are defined differently when background knowledge is incorporated. If words in a test example are found in some background knowledge, then other words that are in that background knowledge can connect this test example to dissimilar (in terms of word overlap and direct cosine difference) training examples. The final classification thus integrates information from multiple training examples and the multiple "bridge" examples that lie between them in the background text.

Note that this approach does not concern itself with which class (if any!) a background item belongs to. We instead simply use the text directly as part of the decision-making process. This is in contrast to an approach that would explicitly use the training set to classify the background items as if they were true examples, and then add them to the labeled set. Our method allows for more sophisticated use and combination of the training instances and background knowledge. A background instance that is close to numerous training instances can be included more than once in the table returned by the WHIRL query – even if the training examples that it is close to have different classes. Similarly, a training example can also be included in the table multiple times, if it is close to numerous background instances. Suppose that our classification task consists of labeling the first few words of a news article with a topic. If a test example belongs to the category *sports*, for instance, the cosine distance between the few words in the test example and each of the small number of training examples might be large. However, we would hope that given a large corpus of unlabeled news articles, it is likely that there will be one or more articles that contains both the few words of the test example and the words of one of the training examples.

Finally, note that, our use of "background knowledge" in a WHIRL query is in a sense a form of query expansion (Buckley et al., 1995). Instead of directly searching for the training examples that are closest to a test example we search for the training examples that are closest to the "background knowledge" expansion of the the test example. However, unlike standard query expansion, and because of our conjunctive conditions, the background knowledge expansion itself is chosen with respect to the training example that it is close to. This means that each query has multiple expansions, and all those that maximize the score of the conjunctive condition are combined.

3. Experiments and Results

We have tested our system on four distinct text-categorization tasks that we have taken from the World Wide Web. In each case, the training and test examples are short text strings, a problem that is prevalent in real-

world applications and for which WHIRL was especially designed. For each of our four problems, the source of our background knowledge varies, sometimes originating at the same site from which we obtained the labeled data, and sometimes from unrelated sites also found on the Web.

3.1 Data Sets

Technical papers One common text categorization task is assigning discipline or sub-discipline names to technical papers. We created a data-set from the physics papers archive (http://xxx.lanl.gov), where we downloaded the titles for all technical papers in the first three areas in physics (astrophysics, condensed matter, and general relativity and quantum cosmology) for the month of March 1999. As background knowledge we downloaded the abstracts of all papers in these same areas from the two previous months – January and February 1999. In total there were 1701 pieces of knowledge in the background set, and 1066 in the training-test set combined. The distribution of classes was skewed, however, as there were 493 titles in astrophysics, 460 in condensed matter, and only 113 in quantum cosmology. These background knowledge abstracts were downloaded without their labels (i.e., without knowledge of what sub-discipline they were from) so that our learning program had no access to them.

News Another data set that we created was obtained from ClariNet news. We downloaded all articles under the sports and banking headings on November 17th 1999, using the most recent ones for training and test sets and the older ones for background knowledge. In total, our background knowledge consisted of a corpus of 1165 articles. The background knowledge in this problem consisted of the first 100 words of each of these articles. Informal studies showed us that including the entire articles did not improve accuracy substantially, and degraded the efficiency of WHIRL. Our training-test data had 1033 data points of which 637 belonged to the sports category, and 406 belonged to banking category. We present four sets of results in connection with the 1033 data points, called 3-words, 5-words, 7-words and 9-words, corresponding to the test and training set consisting of the first 3, 5, 7, or 9 words of each article respectively.

Web page titles To determine the usefulness of WHIRL as a nearest neighbor classification tool, Cohen and Hirsh (1998) used two data sets taken from the World Wide Web. The first, NetVet (http://www.netvet.wustle.edu) included the Web page headings for its pages concerning cows, horses, cats, dogs, rodents, birds and primates. For example, a training example in the class birds might have been: "Wild Bird Center of Walnut Creek". Each of these titles had a URL that linked the title to its associated Web page. For the labeled corpus, we chose half of these titles with their labels, in total 1789 examples. We discarded the other half of the titles, with their labels, and simply kept the URL to the associated Web page. We used these URLs to download the first 100 words from each of these pages, to be placed into a corpus for background knowledge. Those URLs that were not reachable were ignored by the program that created the background knowledge. In total there were 1158 entries in the background knowledge database.

Companies The second of Cohen and Hirsh's data sets consisted of a training set of company names, 2472 in all, taken from the Hoover Web site (http://www.hoovers.com) labeled with one of 124 industry names. We created background knowledge from an entirely different Web site – http://biz.yahoo.com. We downloaded the Web pages under each business category in the Yahoo! business hierarchy to create 101 pieces of background knowledge. The Yahoo! hierarchy had a different number of classes and different way of dividing the companies, but this was irrelevant to our purposes since we treated it solely as a source of unlabeled background text. Each piece of background knowledge consisted of the combination of Web pages that were stored under a sub-topic in the Yahoo! hierarchy. Each instance in the table of background knowledge was thus a much longer text string than the training or test examples.

3.2 Results

We present a series of results and graphs that show that WHIRL incorporating background knowledge most often performs better than WHIRL without the supplementary knowledge. This improvement is most dramatic when there are fewer labeled examples, and when the labeled examples are shorter strings.

However, since a number of the data sets are being used for the first time in this paper, we start this section by comparing the core WHIRL method without background knowledge (which we label WHIRL-nn), to a more traditional method, RIPPER (Cohen 1995), to demonstrate that our improvements are on top of already strong classification performance. Error rates in Table 1 represent average error of five cross-validated runs on the full training set. The "best" value for each problem is shown in bold. Since WHIRL uses the Porter's stemming algorithm when it creates the TFIDF representation, we used it as well, before giving the data to RIPPER. As can be seen from this table, in all data sets, WHIRL-nn outperforms RIPPER, and thus any improvements above and beyond WHIRL-nn that we now report represent even stronger classification performance than this credible state-of-the-art method.

The remainder of our results, presented in a series of figures, report error rates for the baseline WHIRL approach WHIRL-nn to our new approach, which we label WHIRL-bg. In each case we report error rates as we vary the number of training examples given to the learner. Each point

Table 1. Error rates: RIPPER vs WHIRL-nn

Data Set	RIPPER	WHIRL-nn
3-words	20.23	**12.36**
5-words	14.98	**7.11**
7-words	13.07	**5.55**
9-words	11.71	**4.77**
netvet	44.26	**39.20**
hoovers	77.95	**70.90**
2class physics	30.50	**7.89**
3class physics	39.96	**16.90**

represents an average of five cross-validated runs. For each cross-validated run, four-fifths of the data is used as the training set and one-fifth is used as the test set. Holding this test set steady, the number of examples in the training set was varied. Each data-set was tested with both WHIRL-nn and WHIRL-bg using 20, 40, 60, 80, and 100 percent of the data.

We present two sets of results on the physics data in Figure 1 and Figure 2. Figure 1 is a two-class problem, where only the titles of papers in the astrophysics and condensed materials classes were used. These classes had nearly the same number of training examples. Figure 2 is a three class problem, where a class with a fewer number of training examples was added to the previous problem. Figure 1 clearly shows the effect that background knowledge can have on text data sets. The line representing WHIRL-bg remains almost horizontal as fewer training examples were used, indicating that the background knowledge compensated for the lack of data. In contrast, WHIRL-nn sharply degraded as fewer training examples are used. The helpfulness of the background knowledge, therefore, also increased as fewer training examples were used. When the third class was added, error rates of both WHIRL-nn and WHIRL-bg went up. However, the same effect of background knowledge can be seen in Figure 2 as well.

Results on the ClariNet data set are presented in Figures 3–6. The training and test set for these figures vary in terms of number of words in each example, ranging from the first three words in each article, to the first nine words. As expected, the smaller the number of words in each training and test example, the worse both WHIRL-nn and WHIRL-bg performed. The addition of background knowledge was most useful with the shorter strings in the test and training data as well. This is represented in Figures 3–6 by the point at which the two lines intersect. For strings of length 3, background knowledge reduced the error rates, even when the entire set of training data was used. As the number of words in the training-test examples increased, the point at which background knowledge became helpful changed. For strings of length 9, background knowledge reduced er-

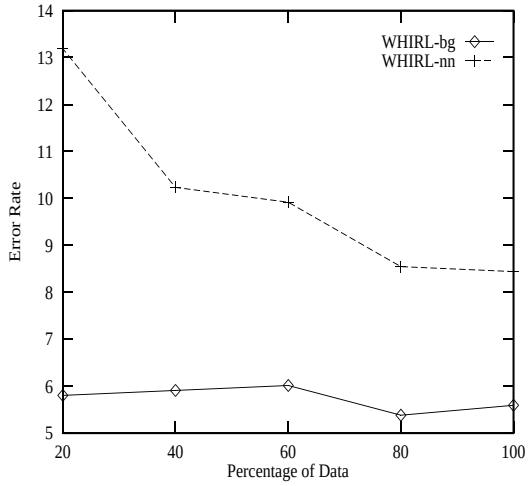

Figure 1. WHIRL-bg and WHIRL-nn for the two class paper title problem

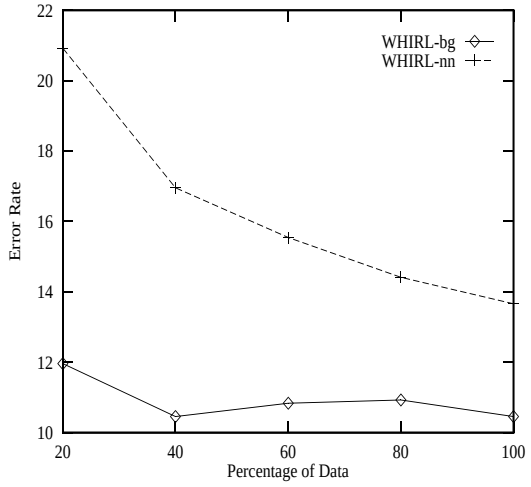

Figure 2. WHIRL-bg and WHIRL-nn for the three class paper title problem

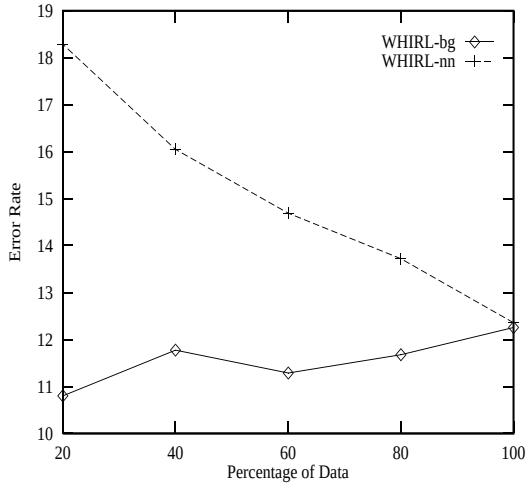

Figure 3. WHIRL-bg and WHIRL-nn for 3-word News

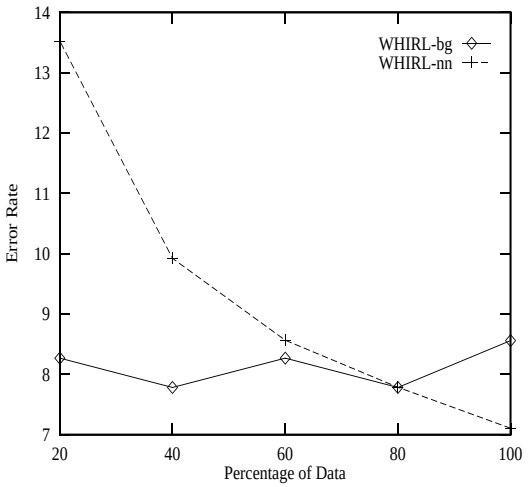

Figure 4. WHIRL-bg and WHIRL-nn for 5-word News

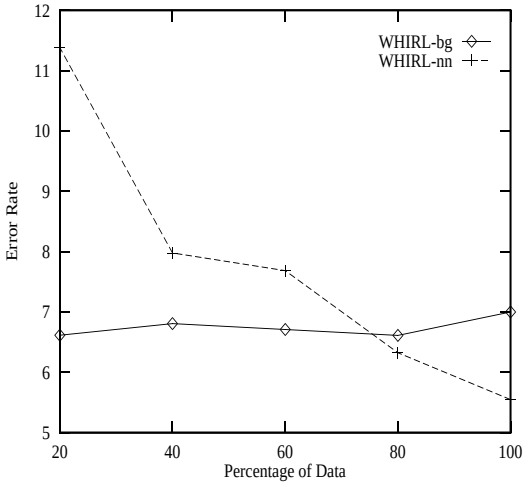

Figure 5. WHIRL-bg and WHIRL-nn for 7-word News

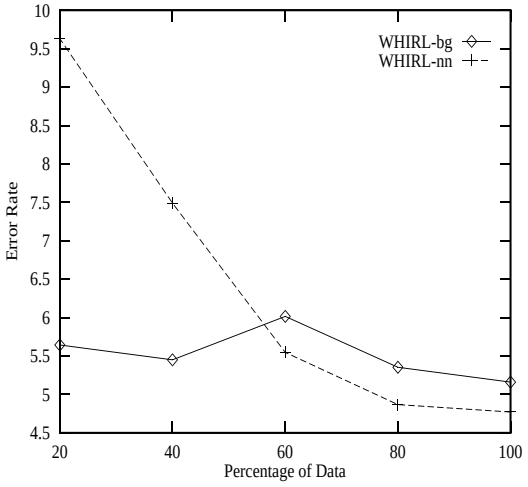

Figure 6. WHIRL-bg and WHIRL-nn for 9-word News

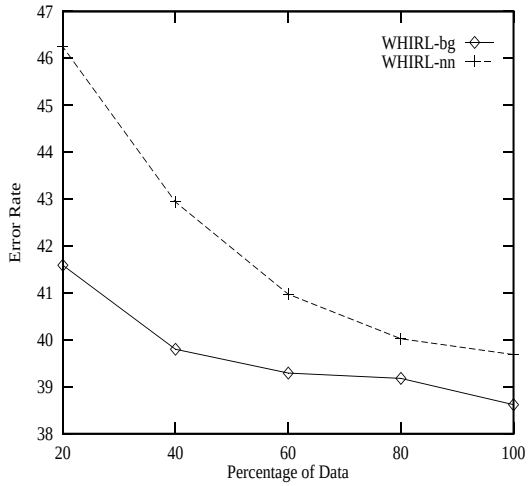

Figure 7. WHIRL-bg and WHIRL-nn for NetVet

ror rates only when less than 60 percent of the data was used. This gives empirical evidence that the less informative the training data is, the greater the advantage in having a corpus of background knowledge available for use during classification. The size of the reduction in error rate obtained by running WHIRL-bg was also greater when there were fewer words in each example.

Results for the NetVet domain are graphed in Figure 7. Reductions in error rate increased as the number of training examples decreased. The NetVet domain is unlike the two previously discussed in that there was overlap in topics in the background knowledge. A Web page that could be useful in classifying a test example as belonging to the category of Dogs was quite likely to discuss Cats and vice versa. Some of the training and test examples, too, could have caused confusion. There were titles of Web pages on pet stores or animal care that were placed under one topic, but could just have easily been placed in many other different categories. We therefore were not surprised to see that the error rate did not decrease by a large percentage.

The results for the Companies data set are graphed in Figure 8. Once again, WHIRL-bg outperformed WHIRL-nn. Using 100 percent of the data, the decrease in error rate is substantial. However, when the percent of training examples that was used is lower, the difference in error rate between the two systems is reduced. This is unlike the results of the previous three domains. This might have been due to the fact that the training and test examples were company names, which often consisted of words that occured only once (for example, *Xerox*) so that reducing the number of training examples actually reduced the dictionary of words in the training corpus substantially. There were therefore fewer words that could be used to find bridges in the background knowledge.

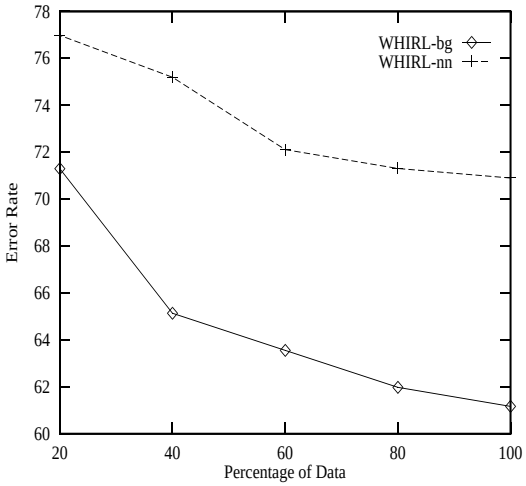

Figure 8. WHIRL-bg and WHIRL-nn for Companies

4. Final Remarks

We have presented a method to reduce error rates in text classification by using a large body of potentially uncoordinated background knowledge. In all four domains to which the system was applied, we saw substantial reductions in error rates, particularly when the set of labeled examples was small. The use of background knowledge allowed for only a small degradation in accuracy in almost all cases, even when only twenty percent of the data was used.

There are a number of limitations of this work that we still have to address. Efficiency of our queries will be an issue that we have to deal with. WHIRL itself is very efficient, yet our approach is essentially a nearest neighbor one, and our "second-order" query with more than one condition has a combinatorial search space.

Since the word combinations in the background knowledge are crucial to the success of this method, another area of future research is the effect of the source and nature of the information in the background knowledge data base. Of the four experimental data sets that we presented, the background knowledge in three of them came from the same site as the training and test sets. Even in our fourth data set, although the background knowledge comes from an unrelated Web site, all the background knowledge is from the same Web site. Ideally, the background knowledge should be culled from a combination of many different sites, and be automatically created through a Web search.

In future work we plan to explore further refinements to the WHIRL-bg approach. We are currently exploring a disjunction of the WHIRL-bg and WHIRL-nn queries. This would allow for correct classification of a test example that is close to the training examples, where appropriate background bridges might not exist. It can also help the issue of efficiency, if useless search is pruned early. We will also begin to look at further extensions of our approach. For example, consider the query:

```
SELECT Test.instance, Train.label
  FROM Train AND Test AND Background as B1
    AND Background as B2
 WHERE Train.instance SIM B1.value
    AND Test.instance SIM B2.value
    AND B1.value SIM B2.value
```

This type of query provides a different way for background knowledge to bridge gaps between a training example and the test example. Given that the test and training examples only have a small bit of knowledge about the class to which they belong, this query allows each small bit of knowledge to be mapped to larger pieces of background knowledge that are similar to each other. We are also investigating the possibility of incorporating weights into the conjunctive clauses of WHIRL to adjust the impact of training and test data similarity.

Acknowledgments

We would like to thank William Cohen for helpful comments and discussions. We would also like to thank the anonymous reviewers for many useful suggestions.

References

Blum, A. & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (pp. 92–100). New York: ACM Press.

Buckley, C., Salton, G., Allan, J., & Singhal, A. (1995). Automatic query expansion using SMART: TREC-3. *Proceedings of the Third Text REtrieval Conference* (pp. 69-80). Gaithersberg, MD: NIST Special Publication 500-225.

Cohen, W. W. (1995). Fast effective rule induction. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 115-123). San Francisco: Morgan Kaufmann.

Cohen, W. W. (1998a). Integration on heterogeneous databases without common domains using queries based on textual similarity. *Proceedings of the 1998 ACM-SIGMOD International Conference on the Management of Data* (pp. 201-212). New York: ACM Press.

Cohen, W. W. (1998b). A web-based information system that reasons with the structured collections of text. *Proceedings of the Second International ACM Conference on Autonomous Agents* (pp. 400-407). New York: ACM Press.

Cohen, W. W. & Hirsh, H. (1998). Joins that generalize: Text categorization using WHIRL. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (pp. 169-173). Menlo Park, California: AAAI Press.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. *Proceedings of the Tenth European Conference on Machine Learning* (pp. 137–142). Berlin: Springer.

Jones, R., McCallum, A., Nigam, K., & Riloff, E. (1999). Bootstrapping for text learning tasks. *Working Notes of the IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications* (pp. 52–63).

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 148–156). San Francisco: Morgan Kaufmann.

Lewis, D. D., & Gale W. (1994). A sequential algorithm for training text classifiers. *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research Development in Information Retrieval* (pp. 3–12). New York: ACM Press.

McCallum, A., & Nigam, K. (1999). Text classification by bootstrapping with keywords, EM and shrinkage. *Working Notes of ACL 1999 Workshop for the Unsupervised Learning in Natural Language Processing* (pp. 52-58)

Mitchell, T. (1999). The role of unlabeled data in supervised learning. *Proceedings of the Sixth International Colloquium on Cognitive Science.*

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (in press). Text classification from labeled and unlabeled documents using EM. *Machine Learning*.

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

Quinlan, J. R. (1993). *C4.5: Programs for machine learning* San Mateo: Morgan Kaufmann.

Salton, G. (1989). Automatic text processing. Reading, MA: Addison-Welsley.

Yang, Y., & Chute, C. (1994). An example-based mapping method for text classification and retrieval. *ACM Transactions on Information Systems, 12, 252-277*.