Joins that Generalize: Text Classification Using WHIRL

William W. Cohen
AT&T Labs—Research
180 Park Avenue
Florham Park NJ 07932
wcohen@research.att.com

Haym Hirsh
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903
hirsh@cs.rutgers.edu

Abstract

WHIRL is an extension of relational databases that can perform “soft joins” based on the similarity of textual identifiers; these soft joins extend the traditional operation of joining tables based on the equivalence of atomic values. This paper evaluates WHIRL on a number of inductive classification tasks using data from the World Wide Web. We show that although WHIRL is designed for more general similarity-based reasoning tasks, it is competitive with mature inductive classification systems on these classification tasks. In particular, WHIRL generally achieves lower generalization error than C4.5, RIPPER, and several nearest-neighbor methods. WHIRL is also fast—up to 500 times faster than C4.5 on some benchmark problems. We also show that WHIRL can be efficiently used to select from a large pool of unlabeled items that can be classified correctly with high confidence.

Introduction

Consider the problem of exploratory analysis of data obtained from the Internet. Assuming that one has already narrowed the set of available information sources to a manageable size, and developed some sort of automatic procedure for extracting data from the sites of interest, what remains is still a difficult problem. There are several operations that one would like to support on the resulting data, including standard relation database management system (DBMS) operations, since some information is in the form of data; text retrieval, since some information is in the form of text; and text categorization. Since data can be collected from many sources, one must also support the (non-standard) DBMS operation of data integration (Monge and Elkan 1997; Hernandez and Stolfo 1995).

As an example of data integration, suppose you have two relations, place(univ, state), which contains university names and the states in which they are located, and job(univ, dept), which contains university departments that are hiring. Suppose further that you are interested in job openings located in a particular state. Normally a user could join the two relations to answer this query. However, what if the relations were extracted from two different and unrelated Web sites? The same university may be referred to as “Rutgers University” in one relation, and “Rutgers, the State University of New Jersey” in another. To solve this problem traditional databases would require some form of key normalization or data cleaning on the relations before they could be joined.

An alternative approach is taken by the database management system (DBMS) WHIRL (Cohen 1998a; 1998b). WHIRL is a conventional DBMS that has been extended to use statistical similarity metrics developed in the information retrieval community to compare and reason about the similarity of pieces of text. These metrics can be used to implement “similarity joins”, an extension of regular joins in which tuples are constructed based on the similarity of values, rather than on equality. Constructed tuples are then presented to the user in an ordered list, with tuples containing the most similar pairs of fields coming first.

As an example, the relations described above could be integrated using the WHIRL query

\[
\text{SELECT place.univ as u1, place.state, job.univ as u2, job.dept} \\
\text{FROM place, job WHERE place.univ~job.univ}
\]

where place.univ~job.univ indicates that these items must be similar. The result of this query is a table with columns u1, state, u2, and dept, and rows sorted by the similarity of u1 and u2. Thus rows in which u1=u2 (i.e., the rows that would be in a standard equijoin of the two relations) will appear first in this table, followed by rows for which u1 and u2 are similar, but not identical (such as the pair “Rutgers University” and “Rutgers, the State University of New Jersey”).

Although WHIRL was designed to query heterogeneous sources of information, WHIRL can also be used in a straightforward manner for traditional text retrieval and relational data operations. Less obviously, WHIRL can be used for inductive classification of text. To see this, note that a simple form of rote learning can be implemented with a conventional DBMS. Assume we are given training data in the form of a relation train(inst, label) associating instances inst with labels label from some fixed set. (For example, instances might be news headlines, and labels might be subject categories from the set {business, sports, other}.) To classify an unlabeled object X one can store X as the only element of a relation test(inst) and use the SQL query:

\[
\text{SELECT test.inst, train.label} \\
\text{FROM train, test WHERE test.inst=train.inst}
\]
In a conventional DBMS this retrieves the correct label for any \( X \) that has been explicitly stored in the training set \( train \). However, if one replaces the equality condition \( test.\text{inst} = train.\text{inst} \) with the similarity condition, \( test.\text{inst} \sim train.\text{inst} \), the resulting WHIRL query will find training instances that are similar to \( X \), and associate their labels with \( X \). More precisely, the answer to the corresponding similarity join will be a table:

<table>
<thead>
<tr>
<th>inst</th>
<th>label</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
<td>( L_1 )</td>
<td>( score_1 )</td>
</tr>
<tr>
<td>( X )</td>
<td>( L_2 )</td>
<td>( score_2 )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Every tuple in this table associates a label \( L_i \) with \( X \). Each \( L_i \) is in the table because there were some instances \( Y_{i_1}, \ldots, Y_{i_k} \) in the training data that had label \( L_i \) and were similar to \( X \). As we will describe below, every tuple in the table also has a “score”, which depends on the number of these \( Y_{j} \)-’s and their similarities to \( X \). This method can thus be viewed as a sort of nearest neighbor classification algorithm.

Elsewhere we have evaluated WHIRL on data integration tasks (Cohen 1998a). In this paper we show that WHIRL is also well-suited to inductive text classification, a task we would also like a general data manipulation system such as WHIRL to support.

**An Overview of WHIRL**

We assume the reader is familiar with relational databases and the SQL query language. WHIRL extends SQL with a new TEXT data type and a similarity operator \( X \sim Y \) that applies to any two items \( X \) and \( Y \) of type TEXT.\(^1\) Associated with this similarity operator is a similarity function \( \text{SIM}(X, Y) \), to be described shortly, whose range is \([0, 1]\), with larger values representing greater similarity.

This paper focuses on queries of the form

\[
\text{SELECT test.\text{inst}, train.label} \\
\text{FROM test.train WHERE test.\text{inst} \sim train.\text{inst}}
\]

and we therefore tailor our explanation to this class of queries. Given a query of this form and a user-specified parameter \( K \), WHIRL will first find the set of \( K \) tuples \((X_i, Y_j, L_j)\) from the Cartesian product of \( test \) and \( train \) such that \( \text{SIM}(X_i, Y_j) \) is largest (where \( L_j \) is \( Y_j \)’s label in \( train \)). Thus, for example, if \( test \) contains a single element \( X \), then the resulting set of tuples corresponds directly to the \( K \) nearest neighbors to \( X \), plus their associated labels. Each of these tuples has a numeric score; in this case the score of \((X_i, Y_j, L_j)\) is simply \( \text{SIM}(X_i, Y_j) \).

The next step for WHIRL is to SELECT \( test.\text{inst}, train.label \), i.e., to select the first and third columns of the table of \((X_i, Y_j, L_j)\) tuples. In this projection step, tuples with equal \( X_i \) and \( L_j \) but different \( Y_j \) will be combined.

The score for each new tuple \((X, L)\) is:

\[
1 - \prod_{j=1}^{n} (1 - p_j)
\]

where \( p_1, \ldots, p_n \) are the scores of the \( n \) tuples \((X, Y_1, L), \ldots, (X, Y_n, L)\) that contain both \( X \) and \( L \).

The similarity of each \( X \) and \( Y \) is assessed via the widely-used “vector space” model of text (Salton 1989). Assume a vocabulary \( T \) of atomic terms that appear in each document.\(^2\) A TEXT object is represented as a vector of real numbers \( v \in \mathbb{R}^{|T|} \), where each component corresponds to a term. We denote with \( v_t \) the component of \( v \) that corresponds to \( t \in T \).

The general idea behind the vector-space representation is that the magnitude of a component \( v_t \) should be related to the “importance” of \( t \) in the document represented by \( v \), with higher weights assigned to terms that are frequent in the document and infrequent in the collection as a whole; the latter often correspond to proper names and other particularly informative terms. We use the TF-IDF weighting scheme (Salton 1989) in which

\[
v_t = \log(TF_{x,t} + 1) \cdot \log(N/IDF_t).
\]

Here \( TF_{x,t} \) is the number of times that \( t \) occurs in the document represented by \( v \), and \( IDF_t = \sum_n N \), where \( N \) is the total number of records, and \( n_t \) is the total number of records that contain the term \( t \) in the given field. The similarity of two document vectors \( v \) and \( w \) is then given by the formula

\[
\text{SIM}(v, w) = \frac{\sum_{t \in T} v_t \cdot w_t}{||v|| \cdot ||w||}
\]

This measure is large if the two vectors share many “important” terms, and small otherwise. Notice that due to the normalizing term \( \text{SIM}(v, w) \) is always between zero and one.

By combining graph search methods with indexing and pruning methods from information retrieval, WHIRL can be implemented fairly efficiently. Details of the implementation and a full description of the semantics of WHIRL can be found elsewhere (Cohen 1998a).

**Experimental Results**

To evaluate WHIRL as a tool for Web-based text classification tasks, we created nine benchmark problems, listed in Table 1, from pages found on the World Wide Web. Although they vary in domain and size, all were generated in a straightforward fashion from the relevant pages, and all associate text — typically short, name-like strings — with labels from a relatively small set.\(^3\) The table gives a summary description of each domain, the number of classes and number of terms found in each problem, and the number of training and testing examples used in our text classification

\(^1\) Other published descriptions used a Prolog-like notation for the same language (Cohen 1998a; 1998b). We refer the reader to these earlier papers for descriptions of the full scope of WHIRL and efficient methods for evaluating general WHIRL queries.

\(^2\) Currently, \( T \) contains all “word stems” (morphologically inspired prefixes) produced by the Porter (1980) stemming algorithm on texts appearing in the database.

\(^3\) Although much work in this area has focused on longer documents, we focus on short text items that more typically appear in such data integration tasks.
experiments. On smaller problems we used 10-fold cross-validation (“10cv”), and on larger problems a single holdout set of the specified size was used. In the case of the cdroms and netvet domains, duplicate instances with different labels were removed.

**Using WHIRL for Inductive Classification**

To label each new test example, the example was placed in a table by itself, and a similarity join was performed with it and the training-set table. This results in a table where the test example is the first item of each tuple, and every label that occurred for any of the top \( K \) most similar text items in the training-set table appears as the second item of some tuple. We used \( K = 30 \), based on exploratory analysis of a small number of datasets, plus the experience of Yang and Chute (1994) with a related classifier (see below). The highest-scoring label was used as the test item’s classification. If the table was empty, the most frequent class was used.

We compared WHIRL to several baseline learning algorithms. C4.5 (Quinlan 1994) is a feature-vector based decision-tree learner. We used as binary-valued features the same set of terms used in WHIRL’s vectors; however, for tractability reasons, only terms that appeared at least 3 times in the training set were used as features. RIPPER (Cohen 1995) is a rule learning system that has been used for text categorization (Cohen and Singer 1996). We used RIPPER’s set-valued features (Cohen 1996) to represent the instances; this is functionally equivalent to using all terms as binary features, but more efficient. No feature selection was performed. 1-NN finds the nearest item in the training-set table (using the vector-space similarity measure) features, but more efficient. No feature selection was performed.

The algorithm that comes closest to WHIRL is \( K\text{-NN}_S \). WHIRL outperforms \( K\text{-NN}_S \) eight of nine times, but all of these eight differences are small. However, five are statistically significant, indicating that WHIRL represents a small but statistically real improvement over \( K\text{-NN}_S \).

**Using WHIRL for Selective Classification**

Although the error rates are still quite high on some tasks, this is not too surprising given the nature of the data. For example, consider the problem of labeling company names with the industry in which they operate (as in hline and hcoarse). It is fairly easy to guess the business of “Watson Pharmaceuticals, Inc.”, but correctly labeling “Adolph Coors Company” or “AT&T” requires domain knowledge.

Of course, in many applications, it is not necessary to label every test instance; often it is enough to find a subset of the test data that can be reliably labeled. For example, in constructing a mailing list, it is not necessary determine if every household in the US is likely to buy the advertised product; one needs only to find a relatively small set of households that are likely customers. The basic problem can be stated as follows: given \( M \) test cases, give labels to \( N \) of these, for \( N \ll M \), where \( N \) is set by the user. We call this problem **selective classification**.

One way to perform selective classification is to use a learner’s hypothesis to make a prediction on each test item, and then pick the \( N \) items predicted with highest confidence. Associating a confidence with a learner’s prediction is usu-
ally straightforward.\(^6\)

However, WHIRL also suggests another approach to selective classification. One can again use the query

```
SELECT test_inst, train, label
FROM train, test
WHERE testInst ~ train
```

but let test contain all unlabeled test cases, rather than a single test case. The result of this query will be a table of tuples \((X_i, L_i)\) where \(X_i\) is a test case and \(L_i\) is a label associated with one or more training instances that are similar to \(X_i\). In implementing this approach we used the score of a tuple (as computed by WHIRL) as a measure of confidence. To avoid returning the same example multiple times with a different labels, we also discarded pairs \((X, L)\) such that \((X, L')\) (with \(L' \neq L\)) also appears in the table with a higher score; this guarantees that each example is labeled at most once.

Rather than uniformly selecting \(K\) nearest neighbors for each test example, this approach (henceforth the “join method”) finds the \(K\) pairs of training and test examples that are most similar, and then projects the final table from this intermediate result. Hence, the join method is not identical to repeating a \(K\)-NN style classification for every test query (for any value of \(K\)).

An advantage of the join method is that since only one query is executed for all the test instances, the join method is simpler and faster than using repeated \(K\)-NN queries. Table 3 compares the efficiency of the baseline learners to the efficiency of the join method. In each case we report the combined time to learn and classify the test cases. The left-hand side of the table gives the absolute runtime for each method, and the right-hand side gives the runtime for the baseline learners as a multiple of the runtime of the join method. The join method is much faster than symbolic methods, with average speedups of nearly 200 for C4.5 — with 500 in the best case — and average speedups of nearly 30 for RIPPER — with 60 in the best case. It is also significantly faster than the straight-forward WHIRL approach, or even using 1-NN on this task.\(^7\) This speedup is obtained even though the size \(K\) of the intermediate table must be fairly large, since it is shared by all test instances. We used \(K = 5000\) in the experiments below, and from preliminary experiments \(K = 1000\) appears to give measurably worse performance.

A potential disadvantage of the join method is that test cases which are not near any training cases will not be given any classification; for the five larger problems, for example, only 1/4 to 1/2 of the test examples receive labels. In selective classification, however, this is not a problem. If one considers only the \(N\) top-scoring classifications for small \(N\), there is no apparent loss in accuracy in using the join method rather than WHIRL; indeed, there appears to be a slight gain. The upper graph in Figure 2 plots the error on the \(N\) selected predictions (also known as the precision at rank \(N\)) against \(N\) for the join method, and compares this to the baseline learners that perform best, according to this metric. All points are averages across all nine datasets. On average, the join method tends to outperform WHIRL for \(N\) less than about 200; after this point WHIRL’s performance begins to dominate. Generally, the join method is very competitive with C4.5 and RIPPER.\(^8\) However, these averages

---

Table 1: Description of benchmark problems

<table>
<thead>
<tr>
<th>problem</th>
<th>#train</th>
<th>#test</th>
<th>#classes</th>
<th>#terms</th>
<th>text-valued field</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>memos</td>
<td>334</td>
<td>10cv</td>
<td>11</td>
<td>1014</td>
<td>document title</td>
<td>category</td>
</tr>
<tr>
<td>cdroms</td>
<td>798</td>
<td>10cv</td>
<td>6</td>
<td>1133</td>
<td>CDRom game name</td>
<td>category</td>
</tr>
<tr>
<td>birdcom</td>
<td>914</td>
<td>10cv</td>
<td>22</td>
<td>674</td>
<td>common name of bird</td>
<td>phylgenic order</td>
</tr>
<tr>
<td>birdsci</td>
<td>914</td>
<td>10cv</td>
<td>22</td>
<td>1738</td>
<td>common + scientific name of bird</td>
<td>phylgenic order</td>
</tr>
<tr>
<td>hcoarse</td>
<td>1875</td>
<td>600</td>
<td>126</td>
<td>2098</td>
<td>company name</td>
<td>industry (coarse grain)</td>
</tr>
<tr>
<td>hfine</td>
<td>1875</td>
<td>600</td>
<td>228</td>
<td>2098</td>
<td>company name</td>
<td>industry (fine grain)</td>
</tr>
<tr>
<td>books</td>
<td>3501</td>
<td>1800</td>
<td>63</td>
<td>7019</td>
<td>book title</td>
<td>subject heading</td>
</tr>
<tr>
<td>species</td>
<td>3119</td>
<td>1600</td>
<td>6</td>
<td>7231</td>
<td>animal name</td>
<td>phylum</td>
</tr>
<tr>
<td>netvet</td>
<td>3596</td>
<td>2000</td>
<td>14</td>
<td>5460</td>
<td>URL title</td>
<td>category</td>
</tr>
</tbody>
</table>

Table 2: Experimental results for accuracy

<table>
<thead>
<tr>
<th>problem</th>
<th>default</th>
<th>RIPPER</th>
<th>C4.5</th>
<th>1-NN</th>
<th>K-NN</th>
<th>K-NN</th>
<th>WHIRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>memos</td>
<td>19.8</td>
<td>30.9</td>
<td>57.3</td>
<td>68.9</td>
<td>51.2</td>
<td>64.4</td>
<td>66.5</td>
</tr>
<tr>
<td>cdroms</td>
<td>26.4</td>
<td>38.3</td>
<td>39.2</td>
<td>40.5</td>
<td>43.0</td>
<td>47.2</td>
<td>47.5</td>
</tr>
<tr>
<td>birdcom</td>
<td>42.8</td>
<td>88.8</td>
<td>79.6</td>
<td>70.4</td>
<td>78.9</td>
<td>82.3</td>
<td>83.9</td>
</tr>
<tr>
<td>birdsci</td>
<td>42.8</td>
<td>91.0</td>
<td>83.3</td>
<td>82.8</td>
<td>81.5</td>
<td>89.2</td>
<td>90.3</td>
</tr>
<tr>
<td>hcoarse</td>
<td>11.3</td>
<td>28.0</td>
<td>30.2</td>
<td>20.2</td>
<td>29.5</td>
<td>32.7</td>
<td>32.9</td>
</tr>
<tr>
<td>hfine</td>
<td>4.4</td>
<td>16.5</td>
<td>17.2</td>
<td>12.7</td>
<td>17.8</td>
<td>21.0</td>
<td>20.8</td>
</tr>
<tr>
<td>books</td>
<td>5.7</td>
<td>42.3</td>
<td>52.2</td>
<td>53.0</td>
<td>54.7</td>
<td>60.1</td>
<td>61.4</td>
</tr>
<tr>
<td>species</td>
<td>51.8</td>
<td>90.6</td>
<td>89.4</td>
<td>87.5</td>
<td>90.7</td>
<td>93.6</td>
<td>94.3</td>
</tr>
<tr>
<td>netvet</td>
<td>22.4</td>
<td>67.1</td>
<td>68.8</td>
<td>50.9</td>
<td>64.8</td>
<td>67.8</td>
<td>68.1</td>
</tr>
<tr>
<td>average</td>
<td>25.24</td>
<td>57.52</td>
<td>57.41</td>
<td>53.76</td>
<td>56.64</td>
<td>62.00</td>
<td>62.90</td>
</tr>
</tbody>
</table>

\(^6\)For C4.5 and RIPPER, a confidence measure can be obtained using the proportion of each class of the training examples that reached the terminal node or rule used for classifying the test case. For nearest-neighbor approaches a confidence measure can be obtained from the score associated with the method’s combination rule.

\(^7\)Runtimes for the \(K\)-NN variants are comparable to WHIRL’s.

\(^8\)RIPPER performs better than the join method, on average, for
Table 3: Run time of selected learning methods

<table>
<thead>
<tr>
<th></th>
<th>RIPPER</th>
<th>C4.5</th>
<th>1-NN</th>
<th>WHIRL</th>
<th>Join</th>
<th>RIPPER</th>
<th>C4.5</th>
<th>1-NN</th>
<th>WHIRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>hcoarse</td>
<td>92.7</td>
<td>51.4</td>
<td>10.0</td>
<td>21.1</td>
<td>3.6</td>
<td>25.8</td>
<td>14.3</td>
<td>2.8</td>
<td>5.9</td>
</tr>
<tr>
<td>hfine</td>
<td>126.0</td>
<td>55.7</td>
<td>10.1</td>
<td>22.0</td>
<td>3.7</td>
<td>34.1</td>
<td>15.1</td>
<td>2.7</td>
<td>5.9</td>
</tr>
<tr>
<td>books</td>
<td>213.4</td>
<td>11437.2</td>
<td>35.3</td>
<td>72.4</td>
<td>23.0</td>
<td>61.2</td>
<td>307.3</td>
<td>3.9</td>
<td>4.8</td>
</tr>
<tr>
<td>species</td>
<td>299.7</td>
<td>1505.6</td>
<td>19.2</td>
<td>23.5</td>
<td>4.9</td>
<td>8.5</td>
<td>109.5</td>
<td>1.6</td>
<td>2.6</td>
</tr>
<tr>
<td>netvet</td>
<td>122.2</td>
<td>1566.1</td>
<td>22.6</td>
<td>37.8</td>
<td>14.3</td>
<td>109.5</td>
<td>1.6</td>
<td>2.6</td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>170.8</td>
<td>2923.2</td>
<td>19.4</td>
<td>35.4</td>
<td>9.9</td>
<td>27.8</td>
<td>188.7</td>
<td>2.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Figure 2: Experimental results for selective classification

conceal a lot of individual variation; while the join method is best on average, there are many individual cases for which its performance is not best. This is shown in the lower scatter plot in Figure 2.9

Summary

WHIRL extends relational databases to reason about the similarity of text-valued fields using information-retrieval technology. This paper evaluates WHIRL on inductive classification tasks, where it can be applied in a very natural way — in fact, a single unlabeled example can be classified using one simple WHIRL query, and a closely related WHIRL query can be used to “selectively classify” a pool of examples. Our experiments show that, despite its greater generality, WHIRL is extremely competitive with state-of-the-art methods for inductive classification. WHIRL is fast, since special indexing methods can be used to find the neighbors of an instance, and no time is spent building a model. WHIRL was also shown to be generally superior in generalization performance to existing methods.

References


