The First International Workshop on Mining Multiple Information Sources

MMIS'07

Held in conjunction with the 13th KDD International Conference, August 12, 2007

KDD 07

Workshop Co-Chairs

Xingquan Zhu

Ruoming Jin

Gagan Agrawal

ISBN: **978-1-59593-840-4**

The 13th International Conference on Knowledge Discovery and Data Mining
San Jose, California • Aug 12-15, 2007

The Association for Computing Machinery, Inc.
1515 Broadway
New York, New York 10036

ISBN: 978-1-59593-840-4

Additional copies may be ordered prepaid from

ACM Order Department	Phone: 1-800-342-6626
P.O. BOX 11405	(U.S.A. and Canada)
Church Street Station	+1-212-626-0500
New York, NY 10286-1405	(All other countries)
	Fax: +1-212-944-1318
	E-mail: acmhelp@acm.org

Preface

Recent developments in storage technology and network architectures have made it possible and affordable for scientific institutes, commercial enterprises, and government agencies to gather and store data from multiple sources. The increasing globalization has also demanded that many business applications involve storing information at geographically distributed locations for analysis. Examples include market basket transaction data from different branches of a wholesale store, data collections of a particular branch in different time periods, census data of different states in a particular year, and data of a certain state in different years. For years, knowledge discovery and data mining (also referred to as KDD) has proven to be crucial for discovering novel and actionable patterns hidden in the data. Discovering patterns from multiple information sources provides a unique way to reveal complex relationships, such as correlations, contrasts, and similarities across multiple collections.

Although the capability of distributed data storage brings us opportunities to improve the quality of data management and decision making, the nature of these distributed data repositories also generates significant challenges for inter-repository pattern discovery. Here, we list three major ones: (1) how to efficiently identify quality knowledge from a single data source, where patterns reveal local knowledge for each particular data repository, commonly referred to as **local patterns**; (2) how to integrate and unify multiple information sources into one single view such that previous unseen patterns can be discovered, commonly referred to as **global patterns**; and (3) how to discover the relationships of the patterns hidden across multiple information sources, where the features of the patterns (such as pattern frequencies and their utilities) across different data repositories define inter-repository relationships, which we refer to as **inter patterns**.

In the past, researchers proposed many approaches to handle multiple information sources, but solutions have been mainly focused on scaling mining algorithms for the discovery of local patterns and global patterns. The aim of this workshop is to bring together data mining experts to revisit the problem of pattern discovery from multiple information sources, and identify and synthesize current needs for such purposes. Representative questions to be addressed include but are not limited to:

- Mining from heterogeneous information sources
- Data integration and data warehousing
- Multiple information sources management
- Local pattern analysis and fusion
- Global pattern synthesizing and assessment
- Inter pattern discovery and comparison
- Distributed and high-performance data mining
- Stream data mining algorithms
- Mining domain specific multiple information sources
- Security and privacy issues in multiple information sources
- Interactive data mining systems

The workshop received 11 submissions, from which 4 regular papers and 3 short papers were selected. We are grateful to all program committee members for their constructive comments and suggestions in organizing the workshop. We thank them for finishing all the reviews in a very short amount of time. We would also like to thank all the authors who submitted their papers to the workshop; we could not make an excellent workshop program without their support. The support of the National Science Foundation of China (#60674109) is acknowledged!

More information about the workshop can be found at: http://www.cse.fau.edu/~xqzhu/mmis/kdd07_mmis.html

August 2007

Xingquan Zhu
Ruoming Jin
Gagan Agrawal

MMIS-2007 Workshop Organizing and Program Committee

Workshop Co-Chairs:

Xingquan Zhu	Florida Atlantic University, USA
Ruoming Jin	Kent State University, USA
Gagan Agrawal	the Ohio-State University, USA

Program Committee:

Henrique Andrade	IBM T.J. Watson, USA
Philip Chan	Florida Institute of Technology, USA
Ian Davidson	SUNY at Albany, USA
Taghi Khoshgoftaar	Florida Atlantic University, USA
Tao Li	Florida International University, USA
Huan Liu	Arizona State University, USA
Gary M. Weiss	Fordham University, USA
Xintao Wu	UNC Charlotte, USA
Shichao Zhang	University of Technology, Sydney
Zhi-hua Zhou	Nanjing University, China

Table of Contents

List of Workshop Papers

A Combinatorial Fusion Method for Feature Mining

Ye Tian Gary Weiss D. Frank Hsu Qiang Ma
Department of Computer and Information Science
Fordham University
441 East Fordham Road
Bronx, NY 10458
{ytian, gweiss, hsu, ma}@cis.fordham.edu

ABSTRACT

This paper demonstrates how methods borrowed from information fusion can improve the performance of a classifier by constructing ("fusing") new features that are combinations of existing numeric features. This work is an example of local pattern analysis and fusion because it identifies potentially useful patterns (i.e., feature combinations) from a single data source. In our work, we fuse features by mapping the numeric values for each feature to a rank and then averaging these ranks. The quality of the fused features is measured with respect to how well they classify minority-class examples, which makes this method especially effective for dealing with data sets that exhibit class imbalance. This paper evaluates our combinatorial feature fusion method on ten data sets, using three learning methods. The results indicate that our method can be quite effective in improving classifier performance, although it seems to improve the performance of some learning methods more than others.

General Terms

Algorithms, Performance, Experimentation

Keywords

Feature construction, classification, class imbalance, information fusion, combinatorial fusion analysis

1. INTRODUCTION

The performance of a classification algorithm is highly dependent on the descriptions associated with each example. For this reason, practitioners typically spend a great deal to time making sure that these descriptions are accurate and capture the key aspects of the data. A good practitioner will choose the features used to describe the data very carefully. However, deciding which information to encode and how to encode it in a feature is quite difficult and the best way to do so depends not only on the domain, but on the learning method. For this reason, there have been a variety of attempts over the years to automate part of this process. This work has had a variety of names over the years (although sometimes the emphasis is different) and has been called constructive induction [13], feature engineering [18], feature construction [6] and feature mining [11]. In this paper we discuss how existing numerical features can be combined, without human effort, in order to improve classification performance. This work can also be considered an example of local pattern analysis and fusion because we identify potentially useful patterns in the data (i.e., feature combinations) from a single data source.

The work described in this paper is notable for several reasons. First, unlike the majority of work in this area, we are specifically concerned with improving the performance of data with substantial class imbalance. Such problems, although quite challenging, are quite common and are typical in domains such as medical diagnosis [7], fraud detection [4], and in predicting equipment failures [20]. Furthermore, there are reasons to believe that this important class of problems has the most to benefit from feature construction, since some learners may not be able to detect subtle patterns that only become apparent when several features are examined together [19]. Our work also differs from other work in that our feature combination operator does not directly use the values of the component features but rather their ranks. This allows us to combine numerical features in a meaningful way, without worrying about issues such as scaling. This approach is particularly appropriate given the increased interest in the use of ranking in the data mining [10] and machine learning communities [5]. Our approach also can be viewed as an extension of work from the information fusion community, since techniques similar to the ones we use in this paper have been used to "fuse" information from disparate sources [9]. The work in this paper can be viewed as a specific type of information fusion, which we refer to as feature fusion (yet another term for feature construction).

We describe our combinatorial feature-fusion method in detail in Section 2 and then describe our experimental methodology in Section 3. Our experiments will evaluate our combinatorial feature-fusion strategy on ten data sets, using three learning methods (naïve Bayes, decision trees, and nearest-neighbor). The results from these experiments are described and analyzed in Section 4. We then discuss related work in Section 5. We finish by discussing our main conclusions and areas for future work in Section 6.

2. COMBINATORIAL FEATURE FUSION

This section describes the basic combinatorial feature-fusion method. We begin by providing some basic background information in Section 2.1. In Section 2.2 we describe our combinatorial feature-fusion algorithm.

2.1 Background

Our combinatorial feature-fusion method constructs new features by combining old features. In Section 2.1.1 we introduce some basic terminology and describe *how* features are combined, or fused. Then in Section 2.1.2 we discuss a variety of strategies for selecting the features to be fused. The algorithm described in Section 2.2 will then describe how the fusion strategy and fusion mechanism can be used to construct a set of features that will often improve classifier performance.

2.1.1 Terminology and Basic Steps

In this section we will use a simple example to help explain the relevant terminology and preliminary steps related to feature fusion. This example will also be used in Section 2.2 to help explain the feature-fusion algorithm. Please note that because our feature-fusion method only works with numeric features, for simplicity we assume all features are numeric. Non-numeric features are not a problem in practice—they simply will be passed to the classifier, unchanged.

A data set is made up of examples, or records, each of which has a fixed number of features. Consistent with previous work on information fusion [9,10], we view the value of a feature as a *score*. Typical examples of scores are a person's salary, a student's exam score, and a baseball pitcher's earned run average. Note that in the first two cases a higher score is desirable and in the last case a lower one is to be preferred.

Table 1 introduces our sample data set. This data set contains eight examples, labeled A-H, with five numeric features, F1-F5, and a binary class variable with values 0 or 1. In this example class 1 is the minority class and comprises 3/8 or 37.5% of the examples.

Table 1: A sample dataset

	F1	F2	F3	F4	F5	Class
A	1	4	3	2	8	1
B	3	3	5	5	4	0
C	5	5	2	6	7	1
D	7	6	15	3	2	0
E	11	13	16	7	14	0
F	15	16	4	13	11	0
G	9	7	14	1	18	1
H	17	15	9	8	3	0

Early in our combinatorial feature-fusion method we replace each score (i.e., feature value) with a rank, where a lower rank is better. We convert each score into a rank using a *rank function*. In this paper the rank function adheres to the standard notion of a rank. We sort the score values for each feature in either increasing or decreasing order and then assign the rank based on this ordering. Table 2 shows the values of the features for the sample data set after the scores have been replaced by ranks. In this case the ranks were assigned after sorting the feature values in increasing order. As a specific example, because the three lowest values for F3 in Table 1 are 2, 3, 4 and these values appear in rows C, A, and F, respectively, the ranks in Table 2 for F2 for records C, A, and F are 1, 2, and 3, respectively.

We determine whether the ranks should be assigned based on increasing or decreasing order of the score values by determining the performance of the feature using both ordering schemes and then we use the ordering that yields the best performance (we describe how to compute a feature's performance shortly). In our method, once the scores are replaced with a rank they are never used again. The rank values are used when combining features and when invoking the learning algorithm (i.e., the rank values are used as the feature values).

Table 2: Sample data set with scores replaced by ranks

	F1	F2	F3	F4	F5
A	1	2	2	2	5
B	2	1	4	4	3
C	3	3	1	5	4
D	4	4	7	3	1
E	6	6	8	6	7
F	7	8	3	8	6
G	5	5	6	1	8
H	8	7	5	7	2

Next we show how to compute the "performance" of a feature. This performance metric essentially measures how well the rank of the feature correlates with the minority-class examples. That is, for a feature, do the examples with a good rank tend to belong to the minority class? We explain how to compute this performance metric using feature F2 from our sample data set. First, we sort the records in the data set by the rank value of F2. The results are shown in Table 3. The performance of F2 is then computed as the fraction of the records at the "top" of the table that belong to the minority class. How many records do we look at? The number of records is based on the percentage of minority-class examples in the training data. In this case 3 of 8 of the training examples (37.5%) belong to the minority class so we look at the top 3 records. In this example that means that the performance of F2 is 2/3, the number of class "1" values in the top 3 records (recall that "1" is the minority-class value). Given this scheme, the best performance value that is achievable is 1.0.

Table 3: Ranked list for F2

	F2 Rank	Class
B	1	0
A	2	**1**
C	3	**1**
D	4	0
G	5	1
E	6	0
H	7	0
F	8	0

We may similarly compute the performances for all of the individual features. For this simple example, F1–F4 all have performances of 2/3 and F5 has a performance of 0. Table 4 shows the performances of each of the original "unfused" features. In this overly simplified example, four of the features all have the same performance.

Table 4: Performance values for original features

Feature	Performance
F1	0.67
F2	0.67
F3	0.67
F4	0.67
F5	0.00

This method is also used to compute the performance of the combined (i.e., fused) features. However, to do this we need to determine the rank of a fused feature, so we can sort the examples by this rank. We compute this using a *rank combination function*. Our rank combination function averages the ranks of the features to be combined. This is done for each record. As an example, suppose we want to fuse features F1–F5 and create a new feature, F1F2F3F4F5, which we will call F6. Table 5 shows the rank values for F6 for all eight records. The value for F6 for record A is computed as: (Rank(F1) + Rank(F2) + Rank(F3) + Rank(F4) + Rank(F5))/5 = (1+2+2+2+5)/5 = 2.4. We see that for this new feature, record "A" has the best (lowest) rank. Given these values, one can now compute the performance of the feature F6. Note that even though the values in Table 5 are not integers, we can still consider them ranks. In order to compute the performance of F6, we only need to be able to sort by these values.

Table 5: Rank values for F6 (F1F2F3F4F5)

	F6		F6
A	2.4	**E**	6.6
B	2.8	**F**	6.4
C	3.2	**G**	5.0
D	3.8	**H**	5.8

2.1.2 Combinatorial Fusion Strategies

The previous section introduced the terminology and basic steps required by our combinatorial fusion algorithm, but did not discuss how we decide *which* features to fuse. We discuss that topic in this section. There are many possible strategies for choosing features to "fuse." In this paper we consider combinatorial strategies that look at all possible combinations. However, because this is generally not feasible, due to the number of features that would be introduced, we consider some more restrictive strategies. Let *n* equal the number of numeric features available for combination. To look at all possible combinations would require that we try each single feature, all pairs of features, all triples, etc. The total number of combinations would therefore equal C(n,1) + C(n,2) + … C(n, n), which equals $2^n - 1$. We refer to such a combinatorial fusion strategy as a fully-exhaustive fusion strategy.

We consider more restrictive variants of the fully-exhaustive fusion strategy because, depending on the value of *n*, this strategy may not be practical. The *k-exhaustive* fusion strategy will create all possible combinations using *k* of the *n* numeric features ($k < n$). For example, a 6-exhaustive strategy for a data set with 20 numeric features will select 6 numeric features and then fuse them in all possible ways. Doing so will reduce the number of feature combinations by a factor of 2^{14}. In our algorithm we choose the subset of *k* features based on the performance values for the features, such as the ones in Table 4. Because it will not be expensive to include all of the original features, we also include the $n - k$ original features not used in the fusion process. The 6-exhaustive fusion strategy is one of the three strategies analyzed in this paper.

The k-exhaustive fusion strategy trades off a reduced number of features for the ability to fully combine these features. In some cases it may be better to involve more features in the fusion process, even if they cannot be fused in all possible ways. The *k-fusion* strategy will use all *n* numeric features, but the length of the fused features is limited to length *k*. Thus if we have a data set with 20 numeric features and employ 2-fusion, all possible combinations of single features and pairs of features will be generated. This would yield C(20,1) + C(20,2) = 20 + 190 = 210 features. Similarly, 3-fusion would consider C(20,1) + C(20, 2) + C(20, 3), or 1140 feature combinations.

Table 6 shows the number of features generated by the different fusion strategies. In all cases, as stated before, all original features are included (there are C(n,1) of these). Note that some cells are empty since $k \leq n$. If $k = n$ then the value computed is displayed in bold and corresponds to the fully-exhaustive strategy. Table 6 demonstrates that, given a limit on the number of features we can evaluate, we have a choice of fusion strategies. For example, given ten numeric features, one can use all ten features and generate combinations of length four, which would generate 385 features, or instead select the seven best ones and then fuse those in all possible ways (i.e., up to length 7), which would generate about 127 features (actually 130 since we would include the three original features which were excluded).

Table 6: Combinatorial fusion table

Number Features	k-fusion for values of k shown below									
	1	2	3	4	5	6	7	8	9	10
1	**1**									
2	2	**3**								
3	3	6	**7**							
4	4	10	14	**15**						
5	5	15	25	30	**31**					
6	6	21	41	56	62	**63**				
7	7	28	63	98	119	126	**127**			
8	8	36	92	162	218	246	254	**255**		
9	9	45	129	255	381	465	501	510	**511**	
10	10	55	175	385	637	847	967	1012	1022	**1023**

2.2 The Combinatorial Fusion Algorithm

We now describe the algorithm for performing the combinatorial fusion. This algorithm is summarized in Table 7. We explain this algorithm by working through a complete example, based on the data set introduced in Table 1 of Section 2.1.

For this example, we will use the 5-exhaustive strategy, so that we select the five best performing features and then fuse them in all possible ways. On line 1 of the algorithm in Table 7 we pass into the Comb-Fusion function the data, the features, a *k* value of 5 and a value of True for the Exhaustive flag. As mentioned previously, the data and features are from Table 1. The next few steps were already described in Section 2.1.1. First we convert the scores to ranks (line 3). We then calculate the performance of each of the original (unfused) features, in the loop from lines 4-6. Then in lines 7-11 we determine which features are available for fusion. Since the Exhaustive flag is set, we restrict ourselves to the *k* best features (otherwise all features are available although they then may not be fused in all possible ways).

Table 7: The feature-fusion algorithm

```
1.  Function Comb-Fusion (Data, Features, k, Exhaustive)
2.  {
3.     ConvertScoresToRanks(Data, Features);
4.     for (f=1, f ≤ length(Features) , f++){
5.         Perf[f]=CalculatePerformance(f);
6.     }
7.     if (Exhaustive == TRUE) {
8.         FeaturesForFusion = best k features from Perf[];
9.     } else {
10.        FeaturesForFusion = Features;
11.    }
12.    New = FuseFeatures(FeaturesForFusion, k, Exhaustive);
13.    for (f=1, f ≤ length(New) , f++){
14.        CalculateRank(f);
15.        Perf2[f]=CalculatePerformance(f);
16.    }
17.    Sort(Perf2);
18.    Candidates = Perf2.features;

19.    // We now build up the final feature set
20.    Keep = Features;        // always use original features
21.    partition(Data, *TrainValid, Test);
22.    for (f in Candidates)
23.    {
24.        for (run=1; run ≤ 10, run++)
25.        {
26.            partition(TrainValid, *Training, *Validation);
27.            classifier = build-classifier(Training, Keep);
28.            PerfWithout[run] = evaluate(classifier, Validation);
29.            cand = pop(Candidates);
30.            classifier=build-classifier(Training, Keep ∪ cand);
31.            PerfWith[run] = evaluate(classifier, Validation);
32.        }
33.        if ( average(PerfWith[ ]) > average(PerfWithout[ ]) )
34.        {
35.            pval = t-test(PerfWith[], PerfWithout[]);
36.            if (pval ≤ .10) {
37.                Keep = Keep ∪ cand;
38.            }
39.        }
40.    }  // end for (f in Candidates)
41.    final-classifier = build-classifier(Training, Keep);
42.    final-performance = evaluate(Test, Keep);
43. } // end Function Comb-Fusion
```

The actual generation of the fused features occurs on line 12. In this case, the five best features in FeaturesForFusion will be combined in all possible ways (in this example there are only five features to begin with). Given our decision to always include the original features to the classifier, the original features need not be returned by FuseFeatures (they are handled later on line 20).

Next, on lines 13-16 we calculate the rank for each fused feature and then calculate its performance. This is essentially the same steps that were done earlier for the original, unfused, features. We then sort the features by decreasing performance value (line 17) and then extract the features from this sorted list and save them (line 18) in Candidates, the ordered list of candidate fused features. The results for our simple example are shown in Table 8. We show only the 14 best performing fused features. In this case Candidates would equal {F3F4, F1F2, F1F3, …}.

Table 8: Performance values for 5-exhaustive strategy

Priority	Feature	Perf.	Priority	Feature	Perf.
1	F3F4	1	8	F1F2F4	0.67
2	F1F2	0.67	9	F1F3F4	0.67
3	F1F3	0.67	10	F1F3F5	0.67
4	F2F3	0.67	11	F2F3F4	0.67
5	F2F4	0.67	12	F3F4F5	0.67
6	F3F5	0.67	13	F1F2F3F4	0.67
7	F1F2F3	0.67	14	F1F2F3F5	0.67

We have now completed the first half of the algorithm. In the second half, starting at line 19, we decide which of the Candidate features to include in the final feature set. We begin by initializing Keep to the set of original features. We then partition the data (line 21) into one set to be used for training and validation and another for testing. Beginning on line 22 we iterate over all of the fused features in the Candidate set.

A key question is how we determine when to add a feature. Even though a feature has a good performance score, it may not be useful. For example, the information encoded in the feature may be redundant with the features already included in the feature set. We adopt a pragmatic approach and only add a feature if it improves classifier performance on the validation set and the improvement is statistically significant. To determine this, within this main loop in the second half of the algorithm (lines 22 – 40) we execute ten runs (lines 24 – 32), repeatedly partitioning the training data into a training set and a validation set (line 26). If, averaged over the 10 runs (line 33) the classifier generated with the candidate feature (line 30) outperforms the classifier generated without it (line 28), and the p-value returned by the t-test (line 35) is ≤ .10 (line 36), then we add the feature to Keep (line 37). A p-value ≤ .10 means that we are 90% confident that the observed improvement reflects a true improvement in performance. In steps 41 and 42 we build the final classifier and evaluate it on the test set.

We should point out a few things. First, the actual implementation is more efficient (although slightly more difficult to describe). In the actual implementation we only need to build one classifier in the main loop, since the classifier from the previous iteration, and its performance, is still available. Similarly, we do not need to rebuild the classifier as indicated on line 41. The performance of the classifier can be measured using either AUC or accuracy, and we use both measures in our experiments.

Table 9 shows the behavior of our simple example as each feature is considered. We only show the performance for the first 3 features. The last column indicates the feature being considered and a "+" indicates that it is added while the lack of this symbol indicates that it is not added because the conditions on lines 33 and 36 are not both satisfied. Each row corresponds to an iteration of the main loop starting at line 22 in the algorithm. The first row is based on the classifier built from the original feature set, containing features F1-F5. Note that the first and third features that are considered are added, because they show an improvement in AUC and the p-value is ≤ .10. As we add features we also measure the performance of each classifier on the test set, although this is not used in any of the decision making. The AUC for the test set at the end is reported, however. If we stopped the algorithm after the three iterations, we can conclude that the performance improved from an AUC of .682 to .774. It is of course critical not to use the test set results to determine whether to add a feature.

Table 9: The execution of the algorithm on a simple example

AUC		p-value	Feature
valid	test		(+ means added)
0.670	0.682	--	{F1,F2,F3,F4,F5}
0.766	0.757	0.001	**+F3F4**
0.731			F1F2
0.771	0.774	0.063	**+F1F3**

3 DESCRIPTION OF EXPERIMENTS

In this section we provide the background necessary to understand our experiments. In Section 3.1 we describe the datasets employed in our empirical study and in Section 3.2 we describe the three learning methods that we utilize. Section 3.3 then describes our experimental methodology.

3.1 Datasets

The ten data sets used in our study are described in Table 10. The first field provides the data set name, the second the percentage of examples belonging to the minority class, the third specifies the final number of features and the last column lists the data set size. The data sets are ordered in terms of decreasing class imbalance.

Table 10: The Data Sets

Dataset Name	% Minority Class	Number Features	Dataset Size
protein+	0.59	14	20,000
letter-a*	3.9	15	20,000
income*+	5.9	12	10,000
stock*+	9.9	27	7,112
hepatitis*	19.8	12	500
physics+	24.9	8	20,000
german*	30.0	19	1,000
crx*+	44.1	5	450
bands*+	42.2	13	538
boa1+	49.8	25	5,000

The data sets come from a few sources. The hepatitis, bands, income and letter-a data sets were obtained from the UCI machine learning repository [14] and the crx data set was provided in the Data directory that came with the C4.5 code. The boa1 data set was obtained from researchers at AT&T and has been used in previous published work. The physics and bio data sets are from the 2004 KDD CUP challenge. The stock data set was provided by New York University's Stern School of Business.

In order to simplify the presentation and the analysis of our results, data sets with more than two classes were mapped to two-class problems. This was accomplished by designating one of the original classes, typically the least frequently occurring class, as the minority class and then mapping the remaining classes into the majority class. The data sets that originally contained more than two classes are identified with an asterisk (*). The letter-a data set was generated from the letter-recognition data set by making the letter "a" the minority class. Because we are only employing feature fusion for the numeric features, we deleted any non-numeric features from the data sets. While this is not necessary, since our method could just ignore the non-numeric fields, we did this so that we could better determine the impact of the feature fusion method. The data sets that had any non-numeric features are identified with a "+".

3.2. Learning Methods

All of the learning methods that we use in this paper come from the WEKA data mining software [12]. The three learning methods that we use are Naïve Bayes, decision trees and 1-nearest neighbor. The decision tree algorithm is called J48 in WEKA and is an implementation of the C4.5 algorithm. The 1-nearest neighbor algorithm is referred to as IB1 in WEKA.

3.3. Experimental Methodology

The experiments in our study apply a combinatorial feature-fusion strategy to each of the ten data sets listed in Table 10 and then record the performance with and without the fusion strategy. This performance is measured in terms of the area under the ROC curve (AUC), because ROC analysis [3] is a more appropriate performance metric than accuracy when there is class imbalance. Nonetheless, we repeat some of our experiments with accuracy as the performance metric, since doing so it quite straightforward and accuracy is still a very commonly used performance metric. The three combinatorial fusion strategies that are evaluated are the 2-fusion, 3-fusion and 6-exhaustive fusion strategies described in Section 2. In this study we utilize the three learning algorithms listed in Section 3.2 in order to see how the feature-fusion method benefits each algorithm. In the algorithm in Table 7 the data is partitioned such that 50% is used for training, 20% for validation, and 30% for testing.

4. RESULTS

In this section we describe our main results. Because we are interested in improving classifier performance on data sets with class imbalance, and because of the known deficiencies with accuracy as a performance metric [16], we use AUC as our main performance measure. These AUC results are summarized in Table 11. The results are presented for ten data sets using the Naïve Bayes, decision tree, and 1-NN learning methods. Three combinatorial fusion strategies are evaluated: 2-Fusion (2-F), 3-fusion (3-F) and 6-Exhaustive (6-EX). The AUC results are presented first without (w/o) and then with (w) the combinatorial fusion strategy. The "diff" column shows the absolute *improvement* in AUC resulting from the combinatorial fusion strategy, with negative values indicating that combinatorial fusion degraded the performance.

Table 11: AUC Improvement with Combinatorial Fusion

		Bayes			Decision Trees			1-NN		
Dataset	Strat.	w/o	w	Diff	w/o	w	Diff	w/o	w	Diff
	2-F		.923	-.020		.752	.256		.663	.164
bio	3-F	.943	.954	.010	.496	.742	.247	.499	.651	.152
	6-EX		.926	-.017		.759	.264		.663	.164
	2-F		.963	.000		.943	.021		.961	.024
letter-a	3-F	.963	.960	-.003	.922	.919	-.003	.937	.937	.000
	6-EX		.962	-.001		.937	.014		.961	.024
	2-F		.901	.000		.736	.241		.612	.020
income	3-F	.901	.897	-.004	.494	.734	.239	.593	.621	.028
	6-EX		.900	-.001		.739	.245		.612	.020
	2-F		.762	.037		.755	.260		.575	.051
stock	3-F	.725	.767	.043	.496	.751	.255	.524	.578	.054
	6-EX		.769	.044		.747	.252		.564	.040
	2-F		.869	.005		.755	.000		.803	-.016
hepatitis	3-F	.864	.868	.004	.755	.759	.004	.819	.826	.007
	6-EX		.864	.000		.760	.005		.821	.002
	2-F		.498	.000		.499	.000		.504	.000
physics	3-F	.498	.506	.008	.499	.499	.000	.504	.495	-.008
	6-EX		.506	.008		.499	.000		.504	.000
	2-F		.751	.011		.609	.118		.607	-.001
german	3-F	.740	.723	-.017	.492	.606	.115	.609	.593	-.015
	6-EX		.736	-.004		.654	.162		.609	.000
	2-F		.762	.000		.646	.000		.653	.014
crx	3-F	.762	.779	.017	.646	.670	.024	.639	.673	.034
	6-EX		.755	-.007		.722	.076		.667	.028
	2-F		.779	.029		.611	.108		.559	-.096
bands	3-F	.750	.747	-.003	.504	.603	.099	.655	.644	-.012
	6-EX		.744	-.006		.580	.076		.655	.000
	2-F		.596	.024		.538	.041		.509	-.005
boa1	3-F	.571	.602	.031	.497	.548	.050	.515	.509	-.006
	6-EX		.589	.018		.553	.056		.509	-.005

The results in Table 11 indicate that the combinatorial feature fusion method is effective. The results appear to be most positive for the decision tree learning method. The overall impact of the methods is shown in Table 12, which summarizes the results of the ten data sets. Table 12 shows the summarized results for each combinatorial fusion strategy and learning method. It displays the average absolute improvement in AUC as well as the win-lose-draw (W-L-D) record over the 10 data sets.

Table 12: Summarized AUC Results for Ten Data Set

	Bayes		DT		1-NN	
Strategy	AUC	W-L-D	AUC	W-L-D	AUC	W-L-D
2-fusion	0.009	5-1-4	0.105	7-0-3	0.016	5-4-1
3-fusion	0.009	6-4-0	0.103	8-1-1	0.023	5-4-1
6-exhaustive	0.003	3-6-1	0.115	9-0-1	0.027	6-1-3

The results form both tables indicate that decision trees benefit most from combinatorial fusion, with the one-nearest neighbor learning method showing the second-best improvement. Because the maximum AUC value is 1.0, even an average improvement of .023 (for 1-NN using the 3-fusion strategy) is quite substantial. We believe that the decision tree algorithm improves the most because it is incapable of learning combinations of numeric features, because it can only examine a single feature at a time (oblique decision trees are not subject to this limitation). That is, decision trees operate by making axis-parallel cuts in the "concept space" and because of this limitation, it cannot correctly learn a simple concept such as $x + y = 1$ (although it can approximate it). However, with our combinatorial feature fusion method, we would expect a traditional decision tree to easily learn such a concept.

The results do not demonstrate that any of the three combinatorial feature-fusion strategies is a clear winner over the other two. The 6-exhaustive strategy does perform the best for decision trees and one-nearest neighbor, but performs the worst for naïve Bayes. Since the 3-fusion strategy subsumes the 2-fusion strategy (i.e., it generates a superset of the features generated by the 2-fusion strategy) it is worthwhile to compare these two strategies. Because the results are quite similar, the best we can say is that the results are comparable. Our individual results do show that with the 3-fusion method some 3-fused features are generated and used in the final feature set, so we can conclude that the strategies do differ in what features are used (the same is true of the 6-exhaustive strategy). The fact that the 2-fusion strategy performs competitively with the others may indicate that in practice most of the potential benefits that one can achieve with our combination operator can be achieved by combining only two features.

We can analyze the results in Table 11 to determine if the combinatorial feature fusion method is most effective for data sets with the highest levels of class imbalance. The first four data sets listed in the table all have less than 10% of the data set belonging to the minority class (i.e., have a class ratio greater than 9:1). Table 13 shows the average AUC performance over just these four data sets.

Table 13: Summarized AUC Results for 4 Skewed Datasets

	Bayes		DT		1-NN	
Strategy	AUC	W-L-D	AUC	W-L-D	AUC	W-L-D
2-fusion	0.004	1-1-2	0.195	4-0-0	0.065	4-0-0
3-fusion	0.012	2-2-0	0.185	3-1-0	0.059	3-0-1
6-exhaustive	0.006	1-3-0	0.194	4-0-0	0.062	4-0-0

The results in Table 13, when compared to Table 12, show that the combinatorial fusion method is substantially more beneficial, when using the decision trees and one-nearest neighbor method, for the most skewed data sets (i.e., the ones with the most class imbalance). Between these two methods, the improvement averages about twice the improvement over the ten data sets. The Bayes method shows an improvement in AUC also, but given the win-loss-tie record this is less convincing. Because of the limited number of datasets analyzed, these results cannot be considered conclusive, but nonetheless are quite suggestive. There are two explanations for the more substantial improvement for the most highly skewed data sets. First, because the performance measure described in Section 2 is based on the correlation between the fused feature and the minority-class examples, the features that are more likely to improve minority-class performance are considered first. This makes them more likely to be added, since adding other features first might obscure these improvements. Secondly, it is often quite difficult to identify "rare cases" and algorithms that look at multiple features in parallel are more likely to find the subtle classification rules that might otherwise get overlooked [19].

Although our primary interest is in improving classifier performance with respect to the area under the ROC curve, our method can be used to improve accuracy as well. We repeated a subset of our experiments, using accuracy instead of AUC to determine whether adding a fused feature improves the performance of the classifier with the required level of statistical confidence. Table 14 provides the results when using the 2-fusion strategy. We did not repeat these experiments for the other two strategies because AUC is our primary measure of interest and because the three strategies appear to perform similarly.

Table 14: Accuracy Results using 2-Fusion Strategy

Dataset	Bayes			Decision Trees			1-NN		
	w/o	w	Diff	w/o	w	Diff	w/o	w	Diff
bio	98.8	98.8	0.0	99.4	99.4	0.0	99.2	99.2	0.0
letter-a	98.4	98.4	0.0	98.6	98.6	0.0	98.9	98.9	0.0
income	92.0	92.0	0.0	94.5	94.5	0.0	92.4	92.4	0.0
stock	80.4	80.4	0.0	90.3	90.3	0.0	86.3	86.3	0.0
hepatitis	84.0	84.0	0.0	86.2	80.7	-5.6	89.3	89.3	0.0
physics	68.9	75.3	6.5	75.1	75.2	0.1	62.6	75.0	12.4
german	73.2	73.2	0.0	69.5	73.0	3.5	68.1	71.6	3.5
crx	70.1	70.1	0.0	60.3	75.1	14.9	60.4	73.6	13.2
bands	67.0	67.0	0.0	61.4	61.4	0.0	65.3	65.3	0.0
boa1	55.0	57.0	2.0	51.0	56.9	6.0	52.6	57.5	5.0

The results in Table 14 indicate that our combinatorial fusion method is also effective for accuracy. While most of the data sets show no improvement, only in one case did the combinatorial fusion strategy lead to a decrease in accuracy. In contrast, in ten cases there was an increase in accuracy. In virtually every case where the accuracy remains the same, the combinatorial fusion strategy did not add any fused features. Similar to what we saw for AUC, the naïve Bayes method shows the least improvement.

5. RELATED WORK

There has been a significant amount of work on feature mining/feature construction and so in this section we only mention some representative work. One way to organize work in this area is by the operator used to combine the features. In our work, for example, numeric features are combined by mapping their feature values (i.e., scores) to ranks and then averaging the values of these ranks.

One approach is to assume that the features represent Boolean values and then use the standard logical operators (e.g., $\wedge$,$\vee$) to combine the features[2]. Other methods, such as the X-of-N method [21] differ in some ways but can nonetheless be used to implement most of the logical operators. These logic-based methods require that categorical features and numerical features first be mapped into Boolean values. This is not necessarily difficult, since data reduction algorithms already exist for this (e.g., C4.5 [17] can generate binary splits for numerical features), but this step loses information and, especially for numerical features, may not be a natural choice. This mapping can also lead to other problems. For example, with decision trees, repeatedly partitioning a numeric feature into binary values can lead to data fragmentation, whereas our method actually reduces this problem by allowing one to combine multiple numeric features directly.

Other methods are much more ambitious in the operators they implement. For example, some systems implement multiple mathematical operators, such as "+", "-", "×", and "÷", and relational operators such as "≤" and "≥" [1, 15]. Because these systems provide a richer set of operators than we do, it is not feasible to try all possible combinations, such as our exhaustive fusion strategies. These methods rely on more complex, search-based, heuristic methods. Thus our method has the advantage of simplicity. Again, a key difference is that our method combines ranks, whereas this other work combines the scores.

It is informative to describe our work using a general framework for feature construction that has been proposed [13]. This framework involves the following four steps: 1) detection of when feature construction is required, 2) selection of constructors, 3) generalization of the selected constructors, and 4) evaluation of the new features. The first step involves detecting when there is a need for feature construction. We use a "no detection" policy since we always perform feature construction. Other options would be to only construct features when the classifier that is initially produced fails to meet a pre-specified requirement (e.g., accuracy is too low). The next step involves selecting the constructors to use, which in our case is simple since we only have a single constructor (i.e., combination operator). We do not generalize during the feature construction process, so our work is *not* an example of constructive induction. Finally, we evaluate our new features by tentatively adding them to the classifier one at a time, based on their ranked performance, but only keep the feature if we are statistically confident that it will improve classifier performance.

Feature selection [8], which involves determining which features are useful and should be selected, is often mentioned in the same context as feature construction. Although we did not discuss feature selection in this paper, we have used the techniques in this paper to implement feature selection and plan to investigate this topic in the future. In particular, the measure of feature performance that is introduced in this paper can be used to select the most useful features and to prune feature with low performance.

6. CONCLUSION

This paper examined how a method from information fusion could be applied to feature construction. This method was described in detail and then three combinatorial fusion strategies were evaluated on ten data sets and three learning methods. This combinatorial feature-fusion method was applied to numeric features and our results were quite positive, especially for the data sets with the greatest class imbalance. When measuring AUC, the methods were of greatest benefit to the decision tree learning method, although it also substantially improved the performance of the 1-nearest neighbor method. Our results also indicate that this method is effective at improving accuracy.

The work described in this can be extended in many ways. While the ten data sets we analyzed is a reasonable number of data sets, it clearly would be a benefit to evaluate our methods on additional data sets, including several additional, highly imbalanced, data sets. It would also be interesting to evaluate additional combinatorial feature-fusion strategies, other than the three we evaluated in this paper. However, we suspect more complex fusion strategies will not yield substantial further improvements, so we do not view this as a critical limitation of our current work.

We also think that our basic algorithm can be extended in a few ways. First, although we rank our features and feature combinations and then evaluate their performance based on their correlation with minority-class examples, these are only used to determine the order in which the fused features are considered for inclusion. We plan on evaluating heuristic methods, which would prune feature combinations that perform poorly. This would enable us to evaluate more complex fusion schemes with similar effort or to improve the computational performance of the algorithm. In this same vein, we also wish to consider simplifying the method for deciding whether to add a feature. Currently we use a validation set and only add a feature if the improvement in performance passes a statistical significance test. While there are benefits to this strategy, it also increases the computational requirements of the algorithm.

Finally, we would like to apply our combinatorial method to numeric features using operators that do more than *average* the ranks of the features. While this would greatly expand the space of possible combinations, it could lead to further improvements.

7. REFERENCES

[1] Bloedorn, E. and Michalski, R.S. Data-driven constructive induction in AQ17-PRE: a method and experiments. *Proceedings of the Third International Conference on Tools*, 1991.

[2] Blum, A. and Langley, P. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, December 1997, 97(1-2):245-271.

[3] Bradley, A. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition, 30, 7*(July 1997), 1145-1159.

[4] Chan, P.K., and Stolfo, S.J. Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (*KDD '98)*, (New York, NY, Aug. 27-31, 1998), AAAI Press, 1998, 2001, 164-168.

[5] Cohen, W., Schapire, R., and Singer, Y. Learning to order things. *Journal of Artificial Intelligence Research*, 10 (1999), 243-270.

[6] Flach, P. and Lavrac, N. The role of feature construction in inductive rule learning. In: *Proceedings of the ICML2000 workshop on Attribute-Value and Relational Learning: crossing the boundaries*, 2000.

[7] Gryzmala-Busse, J. W., Zheng, Z., Goodwin, L. K., and Gryzmala-Busse, W. J. An approach to imbalanced data sets based on changing rule strength. In *Learning from Imbalanced Data Sets: Papers from the AAAI Workshop* (Austin, TX, July 31, 2000), AAAI Press, 2000.

[8] Guyon, I., and Elisseef, A. An introduction to variable and feature selection, *Journal of Machine Learning Research,* 3 (2003), 1157-1182.

[9] Hsu, D.F., Chung, Y. and Kristal, B. Combinatorial fusion analysis: methods and practices of combining multiple scoring systems. *Advanced Data Mining Technologies in Bioinformatics. Hershey, PA: Idea Group Publishing*; 2006, 32–62.

[10] Hsu, D. F., and Taksa, I. Comparing rank and score combination methods for data fusion in information retrieval, *Information Retrieval*, 8, 3 (2005), 449-480.

[11] Ma, C., Zhou, D. and Zhou, Y. Feature mining and integration for improving the prediction accuracy of translation initiation sites in eukaryotic mRNAs. In *Fifth International Conference on Grid and Cooperative Computing Workshops*, 2006, 349-356.

[12] Markov, Z., Russell, I.. An introduction to the WEKA data mining system. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. 2006, 367–368.

[13] Matheus, C.J. and Rendell, L.A. Constructive induction on decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989, 645-650.

[14] Newman, D. J., Hettich, S., Blake, C.L., and Merz, C. J. *UCI repository of machine learning databases* [http://www.ics.usi.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science. 1998.

[15] Otero, F., Silva, M., Freitas, A. and Nievola, J. Genetic programming for attribute construction in data mining. In *Proceedings of 6th European Conference,* April 14-16, 2003.

[16] Provost, F., Fawcett, T., and Kohavi, R. The case against accuracy estimation for comparing classifiers. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)* (Madison, Wisconsin, July 24-27, 1998), Morgan Kaufmann, 1998, 445-453.

[17] Quinlan, J. R. C4.5: Programs for machine learning. Morgann Kaufmann, San Mateo, CA, 1993.

[18] Scott, S. and Matwin, S., Feature engineering for text classification. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999, 379-388.

[19] Weiss, G. M. Mining with rarity: a unifying framework. *SIGKDD Explorations*, *6, 1* (Dec. 2004), 7-19.

[20] Weiss, G. M., and Hirsh, H. Learning to predict rare events in event sequences. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (*KDD '98)*, (New York, NY, Aug. 27-31, 1998), AAAI Press, 1998, 359-363.

[21] Zheng, Z.J. Constructing X-of-N attributes for decision tree learning. *Machine Learning*, 40, 1 (2000), 35-75.

Finding New Customers Using Unstructured and Structured Data

Prem Melville, Yan Liu, Richard Lawrence,
Ildar Khabibrakhmanov, Cezar Pendus, Timothy Bowden
IBM T.J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598
{pmelvil,liuya,ricklawr,ildar,cpendus,trbowden}@us.ibm.com

ABSTRACT

Identifying new customers is a critical task for any sales-oriented company. Of particular interest are companies that sell to other businesses, for which there is a wealth of structured information available through financial and firmographic databases. We demonstrate that the content of company web sites can often be an even richer source of information in identifying particular business alignments. We show how supervised learning can be used to build effective predictive models on unstructured web content as well as on structured firmographic data. We also explore methods to leverage the strengths of both sources by combining these data sources. Extensive empirical evaluation on a real-world marketing case study show promising results of our modeling efforts.

1. INTRODUCTION

Sales-oriented companies must continue to find new customers for their products and service offerings. For companies that sell to other businesses, this means identifying new companies with potential interest in purchasing the seller's offerings. Aside from specific for-purchase marketing databases, there are several sources of data relevant to this task. These include

1. Extensive financial information for publicly-traded companies (*e.g.* Standard and Poor's [7])
2. Firmographic data (*e.g.* location, industry, estimated company revenue and number of employees) for a large number of companies (*e.g.* D&B [1])
3. News feeds (*e.g.* Reuters [5])
4. Content extracted from the websites of a universe of potential customers.

Any of these sources of data can be joined with the seller's historical transactions as a basis for building probability-to-purchase models (*e.g.* [25]). For example, D&B firmographic information can be joined with past transactions to build customer targeting models [18] that estimate purchase probabilities based a labeled set of positive examples, *i.e.* previous purchasers of a specific product. But there can be instances where past transactional data are either unavailable or not immediately relevant to the specific task. For example, suppose we are interested in a slightly different business objective, namely identifying companies with whom we might wish to form a business partnership. Such a partnership could involve an agreement to sell each other's products and/or services. In this case, we may wish to find companies with a specific sales strategy that compliments our own business objectives. While firmographic data like D&B can be used to identify a broad pool of candidate companies, it does not contain specific information on a company's overall business alignment. It is clear that company websites are much more likely to contain the relevant information.

To illustrate the issues, we consider the following specific example. Let us assume we are interested in finding partners to sell a specific financial offering. We believe that companies interested in such an offering may also be interested in purchasing consulting services around Sarbanes-Oxley [6] compliance. One strategy to tap into this market quickly would be to enter into a co-marketing agreement with a company that sells Sarbanes-Oxley (SOX) consulting services. However, if we do a web search for "Sarbanes Oxley", we will find a lot of useful information on this topic, but relatively few companies that sell services related to it. Indeed, our objective is to find not only companies that specialize in SOX, but to find them within a specific firmographic window. We may wish to exclude both very small and very large companies, and hence limit the search only to companies with annual revenue between \$100M and \$1B. We may be interested only in a specific SIC [8] code covering Professional Services. This expanded search requires a fusion of structured (firmographic) and unstructured data (web content).

This scenario introduces some very interesting machine learning issues in the emerging area of analyzing combined structured and unstructured data. It may be possible to have experts inspect websites selected via a firmographic filter, and generate binary labels reflecting the degree of "fit" to the partner qualifications. In this case, we can build supervised models that learn these characteristics, and use the model to

KDD'07, August 12 – 15, 2007, San Jose, California, USA.

score other companies within this firmographic window. In the following section, we describe the data obtained in such a labeling exercise. In Sections 3 and 4, we describe models built using the web content and the firmographic data, respectively. We are particularly interested here in the relative predictive power obtained by combining these data sources – Section 5 describes our current efforts in this area.

2. THE COMPANY IDENTIFICATION TASK

We have been able to develop a labeled data set for supervised learning via the process summarized in Figure 1. As discussed in the previous section, the specific application is to identify companies with whom we may wish to partner in order to market a particular financial offering. The first step is to develop a universe of potential companies, based solely on firmographic data such as a company revenue and SIC industry classification. The revenue window is set to eliminate large companies since we are looking for mid-size companies with whom to partner. Using the US D&B table with approximately 15M company sites, this query yields a D&B subset of approximately 2400 companies.

The next step is to obtain the URL of the website for each company within this firmographic universe. Our D&B table does not contain this information, so we resort to submitting the company name to various search engines and processing the returned results. While this process is quite reliable for larger companies, it can return incorrect results for the set of relatively small companies under consideration here. We applied a set of heuristics to improve the accuracy of the company-to-URL mapping.

The immediate challenge is to assign a binary label to each company that reflects their perceived qualifications as a business partner. These labels were generated by a team of human experts with a detailed understanding of desirable characteristics of a successful partner. Each website within the 2400-company universe was inspected by this team, and positive labels were assigned to companies that appeared to be reasonable candidates, based solely on the experts' judgment. Note that there are no specific terms that were required to be on a site in order for it to be labeled as a positive. Rather, the experts would browse the site, and form an opinion based on a broad sense of the potential match. As a result of this exercise, 179 companies were labeled as positives, with the balance (2262) taken as negatives.

Once the data are available, our next step is to map the task to a machine learning problem. In essence, finding business partners is similar to a retrieval task or recommendation system, which can be treated as a ranking problem. However, since no reliable confidence score can be obtained for each company (even by human experts), we simply view the task as a classification problem with special properties, i.e unbalanced data and features from multiple sources.

3. WEB CONTENT MODELS

There exist many information sources where one can acquire the business profile of a company, such as Hoovers [4], Factiva [2], and Harte-Hanks [3]. However, with the increase of valuable information on the world wide web, we can gather a wealth of information just from the content of company websites. The rich information on a company's website often

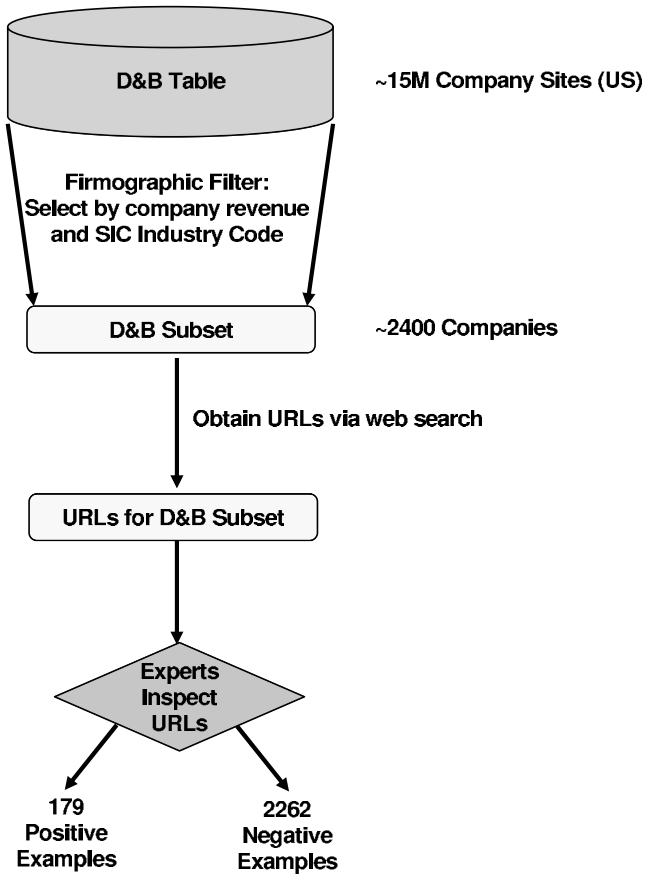

Figure 1: Construction of the labeled data set.

describes the services they support or the products they sell, as well as who their partners are. These are extremely valuable pieces of information that are difficult to acquire from alternative structured sources. In addition, recent changes in the strategies of a company are usually reflected immediately on their websites, while it may take longer for their entries to be updated in databases maintained by third parties.

3.1 Data Preparation

To extract the web content from companies' websites, we first need to find the URLs of the target companies by querying the company names on popular search engines, such as Google or Yahoo!. This seemingly simple task turns out to be rather challenging to automate due to the fact that

1. many of our targets are small or medium-sized companies, and therefore their websites are not ranked at the top in search results;
2. some companies share common words in their names or even share the same name, which makes it difficult to determine the correct URLs, even for humans;
3. some big companies have branches in multiple locations providing different business – in many cases, the search engine will return the URL of their parent company or a wrong branch.

Clearly, the task of automatically identifying URLs for a company itself desires more careful examination. We developed several heuristics to aid in correctly resolving company URLs. We skip the detailed discussion of heuristics here and assume that we have the correct URLs for each company.

Next, for each company in the data set described in Section 2, we crawl the company's website up to a depth of 4 and recompile all the pages into one big file. We pre-process the text by removing stop words, stemming the words into inflected forms (e.g. from the plural form to the singular form and from the past tense to the original form), and selecting features using the χ^2 scores, which is shown to be the best feature-selection method in previous empirical studies [35]. These processes result in a collection with a vocabulary of around 6000 words, which we convert into vectors using the bag-of-word representation with TF-IDF term weighting [12].

3.2 Data Analysis

It is not hard to imagine that there can be some irrelevant information on the company webpages which may affect our modeling results, such as spam advertisements, slogans, contact information, directions, etc.. After some examination, we found that the feature selection algorithm based on χ^2 tests is extremely helpful in reducing such noise in the data. Below is a list of top-ranked words using χ^2 scores:

sarban oxley FDICIA PCAOB outsourc quickbook CPA ERP fraud whitepap firm CFO forens llp financ client payrol sharehold consult COSO

These results are quite encouraging because all these terms have been identified as positively relevant by marketing experts. For example, one type of potential IBM partners are those companies that provide services and consulting on the "Sarbanes-Oxley Act". As a result, the terms such as "PCAOB" and "FDICIA" are relevant because the first refers to a private-sector, non-profit corporation, created by the Sarbanes-Oxley Act, to oversee the auditors of public companies and protect the interest of investigators, while the second term represents the Federal Deposit Insurance Corporation Improvement Act of 1991, which was passed before the Sarbanes-Oxley Act during the savings and loan crisis to strengthen the power of the Federal Deposit Insurance Corporation.

3.3 Experimental Evaluation

Given the web content, we cast the task of customer identification into one of text classification, i.e., given a text document representing a company, classify it as a positive or negative example of a potential customer. We can now use one of many text classification methods available to solve this problem. In particular, we compare the following approaches:

1. SVM-light [17] — an efficient and scalable implementation of Support Vector Machines for text classification.
2. Naïve Bayes using a multinomial text model[19].
3. K-Nearest Neighbor (KNN) [13], with the number of neighbors, k, set to 3.

We also ran versions of the above algorithms modified to deal with the high imbalance between the positive and negative class. SVM-light provides a straightforward mechanism for dealing with class imbalance by specifying a cost factor by which training errors on positive examples outweigh errors on negative examples. We set this cost factor to 10, and refer to this variant as SVM(c=10). As noted by Rennie et al. [24] and Frank and Bouckaert [15] naïve Bayes trained on imbalanced data produces predictions that are biased in favor of large classes. To overcome this, we re-weighted the instances in the training data so that a positive instance has 10 times the weight of a negative instance. We applied the same approach to KNN, which does not affect the choice of neighbors, but influences the relative contribution of positive and negative neighbors in determining a label. We refer to the re-balanced version of naïve Bayes and KNN as naïve Bayes (c=10) and KNN(c=10) respectively.

We compared all methods using 10 fold cross-validation and computed area under the ROC curve (AUC) as the performance metric. Table 1 summarizes the results in terms of AUC, and Figure 2 presents ROC curves comparing different classifiers. For clarity, the figure only shows the classifiers modified for dealing with imbalanced data.

The results show that accounting for the imbalance in data leads to classifiers with better or comparable performance. In particular, correcting for the skewed distributions in naïve Bayes significantly improves its performance, leading to the best classifier for this data. Given that random classification results in an AUC of 0.5, and a perfect classifier results in an AUC of 1, the naïve Bayes AUC score of 0.883 shows that the model is doing extremely well at ordering the instances in terms of likelihood of being a good customer. These results are very encouraging – in the following sections we explore modeling alternative information sources as well as the possibility of improving on the web content models by incorporating information from these sources.

Table 1: Comparing web content models.

Classifier	AUC
Naive Bayes	0.806
Naive Bayes(c=10)	**0.883**
SVM	0.796
SVM(c=10)	0.833
KNN	0.598
KNN(c=10)	0.597

4. FIRMOGRAPHICS MODELS

In the previous section we demonstrated how website content can be effectively used to identify companies that are likely to align well with particular marketing objectives. However, apart from content of company websites, we can also acquire firmographic information about companies through different sources. While web content can be effective in identifying specific sales strategies, firmographic data, such as size and revenue, can be used to identify a broader pool of candidates based on the viability of a sale or collaboration. Typically, firmographics do not contain much specific information about a company's detailed business alignments, however they may still provide valuable information that

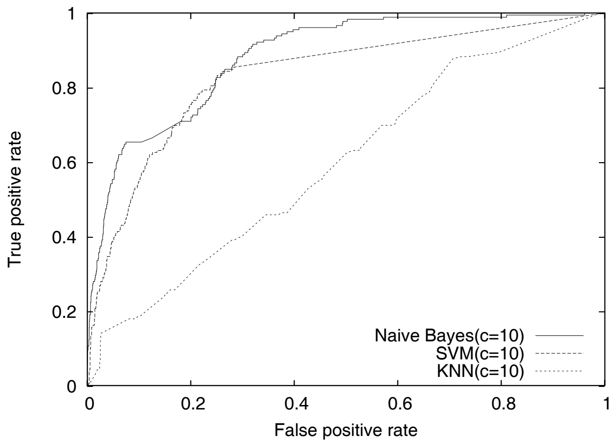

Figure 2: ROC curves comparing web content models.

could be critical in determining potential customers or partners. To verify this, we extracted firmographic data from Dun & Bradstreet (D&B) [1] and IBM Marketing Intelligence [28] which provides information on most businesses in the US and around the world. This data includes information such as:

1. Company size information: revenue and number of employees; along with information on dynamics of change in these figures in recent years.
2. Various levels of industrial classification: industrial sector, industry, sub-industry, etc.
3. Company location: city, state, country.
4. Company structure descriptors: legal status, location in corporate hierarchy (headquarters/subsidiary), etc.

Each company in this data is described by 231 features. We begin by eliminating features that are specific to a company, such as DUNS numbers and names. For ease of modeling, we also exclude categorical features that have too many distinct values, such as zip codes and phone area codes. We remove redundant features that are highly correlated with other features, such as city name and city code. We also eliminate features that are uncorrelated with the target class and are also unlikely to have a causal link with the class label, such as the indicator of a recent address change. Finally, we filter out features that have too many missing values. After these preprocessing steps we are left with 34 features, which are listed in Table 2 in decreasing order of information gain.

We observe that NAICS and SIC codes rank highest on the list of features – this is presumably because they provide the most information regarding business alignments. It is interesting to note that the size of company in terms of number of employees appears to be more important than sales. Furthermore, growth indicators, such as the increase in number of employees and sales in recent years, are far more indicative of a company's classification than the absolute national or global sales. This is consistent with the fact that for the offering under consideration in this data, we are looking for partners that are mid-sized business with the potential to grow.

4.1 Experimental Evaluation

Using the firmographic features listed in Table 2, along with the class labels as before, we compare the following three modeling techniques:

1. J48 [30] — a Java implementation of the C4.5 decision tree algorithm [23].
2. The naïve Bayes algorithm [20], using the Fayyad and Irani approach to discretization [14] of continuous features.
3. Boosted decision stumps, using AdaBoost [26] run for 100 boosting iterations.

To deal with the high imbalance in classes, we re-balance the training data by weighting positive instances 10 times higher than negative instances. Without this re-weighting, decision tree induction (J48) results in a trivial tree with a single leaf node classifying all instances as negative. Figure 3 shows the comparative performance of the different classifiers. The results are summarized in Table 3 in terms of area under the ROC curves. As before, all results were averaged using 10 fold cross-validation.

Boosted decision stumps emerge as being clearly the best approach for this data. These models based solely on firmographics perform surprisingly well at identifying potential customers. However, in absolute terms the firmographics by themselves are not as effective as using the web content (as can be seen by comparing Tables 1 and 3).

Firmographic data provides information that helps identify higher-level characteristics of potential customers, e.g. mid-sized businesses that have been steadily growing. By exploiting industry categorization, the firmographic models can also identify business alignments at a coarse level. For example, the first decision stump in the AdaBoost model learns to classify companies with a NAICS code of 541211 as a positive. This NAICS code corresponds to offices of Certified Public Accountants, which comprises establishments of accountants that are certified to audit the accounting records of public and private organizations and to attest to compliance with generally accepted accounting practices [9]. As described before, in order to market the specific financial service offering for which our data set was created, we are very interested in firms that provide such accounting services. However, in order to be able to further refine our search among all companies within these broad industry classifications, it is crucial that we know the specific services they offer – this is the information we extract from company websites, as done in Section 3.

5. COMBINING INFORMATION FROM MULTIPLE SOURCES

In Sections 3 and 4 we evaluated models built using only web content and firmographics, respectively. The fact that

Table 2: Firmographic features used for modeling, ordered by decreasing information gain.

Feature	Description
NAICS CD	6 digit No. American Industrial Classification Sys
SIC	Standard Industrial Classification code
OFFICE SUPPLY RANK	Based on wholesale buying index. Score 1-100
ST PROVINCE	Name of state or province in which site is located
EMPL RANGE CD	Establishment employee size code
EMPL	Establishment employee size
PC ESTIMATED QTY	Estimates number of PCs at a site
EMPL 5YR PCT	Percent growth in employees (5 year)
CUST PROSPECT CD	Customer or Prospect indicator
WEB PRESENCE CD	Indicates probability of having a Web presence
SALES RNGE CD	Indicates the sales volume range
EMPL 3YR PCT	Percent growth in employees (3 year)
URL STATUS CD	Indicates status of URL for business
NETWORK PC RNGE CD	Estimated number of Nodes or Network connected PCs
STRUCTURE CD	Code for type of business at location
SLE 3YR PCT	Percent growth in sales (3 year)
TECH DEMAND CD	Estimated demand for Technology and Office products
IT BUDGET CD	A ranking of businesses by their likely IT spend
YEAR OWNER CHANGED	Year new owner acquired firm
PTB UNIX SERVER CD	Propensity to buy UNIX servers
YEAR STARTED	Year the Company was Established
SUBSIDIARY CD	Indicates if business is a subsidiary
PTB OTHR SERVER CD	Propensity to buy other servers
WAN PRESENCE CD	Estimated probability of presence of WAN
OFFICE SUPPLY TIER	Ranking of potential to purchase office supplies
PTB WIN SERVER CD	Propensity to buy Windows servers
PUBLIC COMPNY INDC	Indicates if Company is publicly held
SMALL BUSINES INDC	Indicates if enterprise is a small business
NTWK PRESENCE CD	Likely presence of network indicator
WOMEN OWN INDC	Indicates if business is controlled by women
GU SALES US CRCY	Sales for Global Ultimate in whole US dollars
SLE 5YR PCT	Percent growth in sales (5 year)
SALES US CURRENCY	Sales expressed in whole U.S. dollars

Table 3: Comparing firmographics models.

Classifier	AUC
AdaBoost	**0.749**
Naive Bayes	0.692
J48	0.566

it is possible to build effective customer-identification models using each source independently raises the question of whether we can build an even better model by combining these information sources. Although web content is a richer source of information for the task at hand, it is also more susceptible to noise. Automatically mapping company names to their correct URLs in itself is a non-trivial task, and is not 100% accurate. Even with the correct URLs, we end up with lot of noisy (irrelevant) information from company websites such as advertisements, slogans, contact information, etc. On the other hand, firmographic data is more reliable since it is structured and comes directly from database lookups. Hence, web-content and firmographics can be viewed as complimentary sources of information, and by combining them we may be able to leverage the strengths of both sources.

Common approaches to combining information sources vary from early fusion [27], which merges the feature sets extracted from different sources, to late fusion, which combines the output of classifiers built on each features set separately [31]. Following these approaches we explore the following fusion methods:

1. Boosting decision stumps applied to training instances created by merging the web content and firmographic feature sets, which we refer to as AdaBoost (early fusion).

2. Vote-Avg: Build separate classifiers on the web and firmographic features, and average the class probability estimates output by both. We tried two variants of this method using naïve Bayes and SVMs for the web model. In both cases we use AdaBoost for the firmographic models. We refer to the two variants as Vote-Avg (Naive Bayes+AdaBoost) and Vote-Avg (SVM+AdaBoost) respectively.

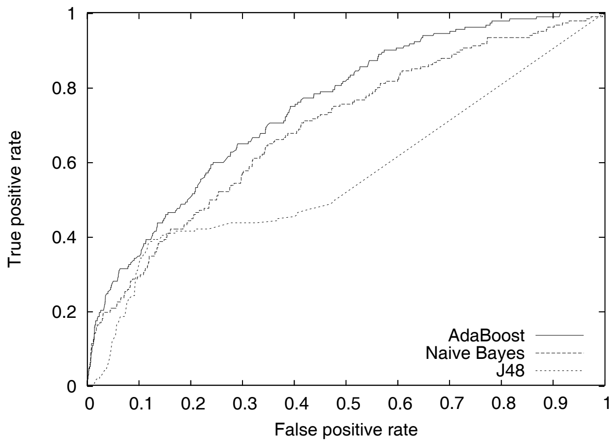

Figure 3: ROC curves comparing firmographics models.

3. Vote-Prod: Follow the same procedure as Vote-Avg, except the class probability estimates produced by individual models are multiplied and renormalized, instead of being averaged. Here again we used the same base model combinations are in Vote-Avg, and refer to them as Vote-Prod (Naive Bayes+AdaBoost) and Vote-Prod (SVM+AdaBoost).

The methods described here can be viewed as *weak* fusion approaches. Below we describe a *strongly-coupled* fusion approach, where the classifiers trained on separate information sources can influence the inductions on each others.

5.1 Transductive Co-training

Given the properties of the data (*i.e.* there are few labeled positive examples, and we have two sources of information available, each of which provides redundant but complementary information), it is natural to apply the co-training algorithm [11]. Originally co-training was developed for semi-supervised learning, which makes use of the unlabeled examples with two distinct sets of features. Following the same idea, we use co-training in the transductive setting, i.e., taking the test set into account during induction and trying to minimize misclassifications of just those particular examples. The details of transductive co-training is shown in Algorithm 1.

The basic assumption of the co-training algorithm is that either set of the features should be sufficient for learning if we had enough labeled data. In our application, this assumption has been obviously violated since there is a lot of noisy or irrelevant information in the web content, while the D&B features do not provide enough information to satisfy the condition. Therefore we do not expect dramatic performance improvement compared with the classifiers using individual sets of features.

As in the case with late fusion, we tried two variants of co-training: naïve Bayes and SVMs for the web model, with AdaBoost for the firmographic models. We refer to these as Co-training (Naive Bayes+Adaboost) and Co-training (SVM+Adaboost), respectively. We use $p = 2$ and $n = 20$

Algorithm 1 Transductive Co-training

```
Given:
$L$ : a set of training examples
$T$: a set of testing examples
$I_{\max}$: maximum iterations

  1. Loop until $T = \emptyset$ or for $I_{\max}$ iterations
  2.     Use $L$ to train a classifier $h_1$ that uses only the
         web content features
  3.     Use $L$ to train a classifier $h_2$ that uses only the
         firmographic features
  4.     Allow $h_1$ to label $p$ most-confident positive and
         $n$ negative examples from $T$
  5.     Allow $h_2$ to label $p$ most-confident positive and
         $n$ negative examples from $T$
  6.     Add these self-labeled examples to $L$
```

in Algorithm 1, in keeping with the low ratio of positives and negatives in the data.

5.2 Experimental Evaluation

The results of all the fusion approaches are summarized in Table 4, and Figure 4 presents ROC curves of different combination techniques. For the sake of clarity we only present ROC curves for three approaches — one each demonstrating early fusion, late fusion and transductive co-training.

Table 4: Comparing methods to combine multiple information sources.

Classifier	AUC
AdaBoost(early fusion)	0.867
Vote-Avg(Naive Bayes+AdaBoost)	0.883
Vote-Prod(Naive Bayes+AdaBoost)	**0.887**
Vote-Avg(SVM+AdaBoost)	0.842
Vote-Prod(SVM+AdaBoost)	0.843
Co-training(Naive Bayes+Adaboost)	0.874
Co-training(SVM+Adaboost)	0.828

Compared with previous approaches using SVMs for the web model (AUC = 0.833, Table 1) and AdaBoost for the firmographic model (AUC = 0.749, Table 3), both early fusion using AdaBoost, as well as both variants of late fusion through voting are successful in improving on the individual models. However, when we use naïve Bayes for the web model (AUC = 0.883, Table 1), only voting with taking products of probabilities (AUC = 0.887) performs better than using only the web content. Furthermore, the added benefit in performance is fairly small.

Taken independently, the web content and firmographic information both lead to useful models for our specific task. However, it also appears that the predictive power realized via the firmographic data can be achieved independently using only the web content. On the other hand, there are also the cases where firmographics helps to correct the ordering of instances of the web models, hence giving rise to the

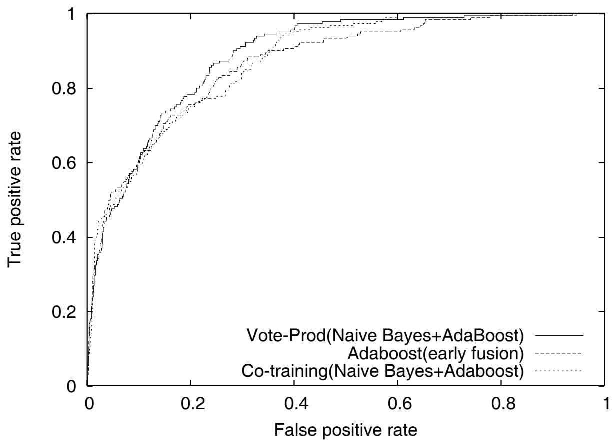

Figure 4: ROC curves comparing combination techniques.

increase, though small, in the area under the ROC curve. Given the high AUC we obtain with the naïve Bayes models on the web content, it is also possible that we are experiencing a ceiling effect, and there is little room for further improvement.

The models built through co-training improve on the firmographic models, but still under-perform the naïve Bayes web models. Blum and Mitchell [11] prove that co-training can learn from unlabeled data starting with only weak classifiers. However, their theoretical guarantees rely on two fundamental assumptions. The first assumption is that the distribution of instances is *compatible* with the target function, i.e., for most instances, the target functions over each feature set predict the same label. In our domain, this means that the class of a company should be identifiable using either the web data or the firmographic data alone. This is clearly not the case, especially since the firmographic data do not provide as much detail in terms of sales strategies of a company. The second assumption is that the two feature sets are conditionally independent of each other, given the class of the instance. This implies that words on a companies web page are not related to the industry classification, and other such firmographic features. This assumption too seems unlikely to hold in practice. Though co-training often works despite the violation of its underlying assumptions [11, 21], it appears that, for this particular domain, it is not as effective, at least in the transductive setting. Given a large pool of unlabeled examples, which is the more common setting for co-training, we may observe better performance.

6. RELATED WORK

In addition to our task in marketing intelligence, there have been many applications in machine learning that share the challenges of combining the information from multiple sources. These include multimedia content analysis from images and text, protein-protein interaction prediction using micro-array data, function annotation as well as sequence information, and sensor networks by combining the data from multiple sensors.

Up to now, the strategies to make use of the information from diverse sources can be summarized as two general approaches, i.e. early fusion, which merges the feature vectors extracted from different data modalities, and late fusion, which combines the output of classifiers built on each single sources [16, 34, 31]. It remains an open question as to which fusion strategy is more appropriate for a certain task, and several comparison studies are discussed for applications in different domains [27, 34]. One extension of the early fusion approach is to derive the latent semantic representation of the data by jointly modeling the low-level features from multiple sources. Possible approaches range from simple methods, such as principal component analysis (PCA), independent component analysis (ICA), Fisher linear discriminant (FLD) and kernel methods, to more sophisticated modeling using graphical models, such as Bayesian model for gene function prediction [29], correspondence latent Dirichlet allocation (Corr-LDA) [10] and dual-wing harmonium models [32] for multimedia applications. These methods have been demonstrated to be more effective than naively joining the low-level features into common feature space. On the other hand, the algorithms in the late fusion approach vary from the equal-weight combination of the sub-classifiers or sub-models, to varied weight with weights learned from cross-validation, and to more adaptive methods with weights depending on the specific testing example [33, 22].

7. CONCLUSION

This paper presents the task of customer identification for companies that sell to other businesses. We formulate this task as a supervised learning problem, and present a case study on an expert-created data set for identifying companies with whom we may wish to partner in order to market a particular financial offering. We analyze the web pages of candidate companies and find that they provide a rich source of information. We demonstrate how we can build highly effective customer-identification models using only this freely available unstructured web content. We also show that, alternatively, we can build models for the same task using data from more structured firmographic information sources. Using firmographics alone can lead to good models, based on coarse-grained characteristics such as industry classifications and dynamics of revenue and employee sizes. However, web content models are more effective because of the richer information available in terms of a company's services, products and sales strategies. Finally, we have explored several approaches to combining the unstructured web content with the structured firmographics data. The results show that by voting classifiers built on the two sources separately, we can get an improvement, albeit small, in the model performance. More sophisticated feature-fusion approaches, such as dual-wing harmonium models [32] may yield better results and provide an avenue for future work.

8. REFERENCES

[1] *Dun and Bradstreet (D&B).* http://www.dnb.com.
[2] *Factiva Dow Jones & Company.* http://www.factiva.com.
[3] *Harte-Hanks Inc.* http://www.harte-hanks.com.
[4] *Hoover's, Inc.* http://www.hoovers.com.
[5] *Reuters.* http://www.reuters.com/.
[6] *Sarbanes-Oxley Act.* http://www.soxlaw.com/.

[7] *Standard & Poor's.* http://www.standardandpoors.com.

[8] *Standard Industrial Classification (SIC).* http://www.sec.gov/info/edgar/siccodes.htm.

[9] *U.S. Census Bureau.* http://www.census.gov.

[10] D. M. Blei and M. I. Jordan. Modeling annotated data. In *SIGIR '03: Proceedings of the 26th annual Int'l ACM SIGIR Conf. on Research and development in information retrieval*, pages 127–134, New York, NY, USA, 2003. ACM Press.

[11] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, Madison, WI, 1998.

[12] C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the seventeenth annual international ACM-SIGIR conference on research and development in information retrieval*. Springer-Verlag, 1994.

[13] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, second edition, 2001.

[14] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1027, Chambéry, France, 1993. Morgan Kaufmann.

[15] E. Frank and R. R. Bouckaert. Naive bayes for text classification with unbalanced classes. In *Proc 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 503–510, 2006.

[16] G. Iyengar and H. J. Nock. Discriminative model fusion for semantic concept detection and annotation in video. In *Proc. of the 11th ACM Int'l Conf. on Multimedia*, pages 255–258, New York, NY, USA, 2003. ACM Press.

[17] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the Tenth European Conference on Machine Learning (ECML-98)*, pages 137–142, Berlin, 1998. Springer-Verlag.

[18] R. Lawrence, C. Perlich, S. Rosset, J. Arroyo, M. Callahan, M. Collins, A. Ershov, S. Feinzig, I. Khabibrakhmanov, S. Mahatma, M. Niemaszyk, and S. Weiss. Analytics-driven solutions for customer targeting and sales force allocation. *IBM Systems Journal*, 2007.

[19] A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *Papers from the AAAI-98 Workshop on Text Categorization*, pages 41–48, Madison, WI, July 1998.

[20] T. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997.

[21] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM-2000)*, pages 86–93, 2000.

[22] W. S. Noble and A. Ben-Hur. Integrating information for protein function prediction. *Bioinformatics - From Genomes to Therapies*, 3, 2007.

[23] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo,CA, 1993.

[24] J. Rennie, L. Shih, J. Teevan, and D. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, pages 616–623, 2003.

[25] S. Rosset and R. Lawrence. Data enhanced predictive modeling for sales targeting. In *Proceedings of SIAM Conference On Data Mining*, 2006.

[26] R. E. Schapire. Theoretical views of boosting and applications. In *Proceedings of the Tenth International Conference on Algorithmic Learning Theory*, pages 13–25, 1999.

[27] C. G. M. Snoek, M. Worring, and A. W. M. Smeulders. Early versus late fusion in semantic video analysis. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402, 2005.

[28] Z. Su, J. Jiang, T. Liu, G. Xie, and Y. Pan. Market intelligence portal: an entity-based system for managing market intelligence. *IBM Systems Journal*, 43(3), 2004.

[29] O. Troyanskaya, K. Dolinski, A. Owen, R. Altman, and D. Botstein. A bayesian framework for combining heterogeneous data sources for gene function prediction (in saccharomyces cerevisiae). *Proc Natl Acad Sci*, 100, 2003.

[30] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, 1999.

[31] Y. Wu, E. Y. Chang, K. C.-C. Chang, and J. R. Smith. Optimal multimodal fusion for multimedia data analysis. In *Proc. of the 12th annual ACM Int'l Conf. on Multimedia*, pages 572–579, 2004.

[32] E. Xing, R. Yan, and A. Hauptmann. Mining associated text and images with dual-wing harmoniums. In *Proceedings of the 21th Annual Conf. on Uncertainty in Artificial Intelligence (UAI-05)*. AUAI press, 2005.

[33] R. Yan. *Probabilistic Models for Combining Diverse Knowledge Sources in Multimedia Retrieval*. PhD thesis, Carnegiel Mellon University, 2006.

[34] R. Yan, J. Yang, and A. G. Hauptmann. Learning query-class dependent weights in automatic video retrieval. In *Proc. of the 12th ACM Int'l Conf. on Multimedia*, pages 548–555. ACM Press, 2004.

[35] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann Publishers, San Francisco, US, 1997.

Incorporating Background Knowledge from the World Wide Web for Rule Evaluation using the Minimum Discriminative Information Principle

Samah Jamal Fodeh
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI, 48824-1226
fodehsam@msu.edu

Pang-Ning Tan
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI, 48824-1226
ptan@msu.edu

ABSTRACT

Discovery of association rules from large transaction databases is a rewarding but challenging data mining endeavor due to the difficulty in distinguishing obvious rules (i.e., rules that are known to the domain experts) from unexpected ones. In this paper, we present a methodology for augmenting background information from an authoritative source on the World Wide Web to the subjective evaluation of association rules. Specifically, we consider the problem of mining association rules in the medical domain, where background information automatically acquired from the MEDLINE database of biomedical citations is used to evaluate the quality of association rules extracted from an electronic medical records (EMR) database. We investigate two approaches for utilizing the rule statistics information obtained from MEDLINE. The first approach detects obvious rules by identifying those with high support or high confidence in the MEDLINE database. The second approach uses the Minimum Discriminative Information (MDI) principle to calibrate the rule confidence, taking into account the reliability of each information source (EMR and MEDLINE).

Keywords

Association Rules, Rule Evaluation, Maximum Entropy.

1. INTRODUCTION

The problem of mining and aggregating information from heterogeneous sources has attracted considerable attention among data mining researchers in recent years. The impetus behind such research is the availability of multiple information sources that can be utilized to enrich the quality of decision making. Mining information from the multiple sources is indeed a very challenging task since each source can have its own structure and quality of information. While there have been considerable efforts devoted toward the problem of synthesizing locally extracted patterns into global patterns, research on enhancing the quality of extracted patterns by incorporating background information from other external sources is still lacking.

Discovery of association rules [1] from large databases has been the subject of extensive research for many years. Association rules not only provide a descriptive explanation of the relationships present in data, they can also be used to build classification [7] and regression models [13]. Despite their usefulness, one of the main practical challenges in association rule mining is how to distinguish interesting rules from spurious ones. Although there have been numerous methods developed for evaluating the statistical significance of association rules [17,10,4,20,5], these methods are often insufficient to isolate rules that are subjectively interesting [16] from those that are obvious to the domain experts.

In this work, we focus on the problem of evaluating the subjective interestingness of association rules by integrating background information from external sources. More specifically, we apply standard mining algorithms such as Apriori [1] to extract statistically significant association rules from a given database and then automatically gather the relevant domain information to help in the interpretation and assessment of the extracted rules. Acquiring the necessary background information from domain experts is often a laborious task. The acquired information is also limited in amount and tends to be static. To overcome these difficulties, we present a methodology for incorporating background information from an authoritative information source on the World Wide Web to enhance the subjective evaluation of association rules. We illustrate the applicability of our proposed methodology in the area of medical informatics, where association rules are first extracted from an electronic medical records (EMR) database and then augmented with information obtained from the PubMed search engine [23], which is a Web service provided by the National Library of Medicine (NLM) to query the MEDLINE database of biomedical citations.

We investigate two approaches for utilizing the rule statistics information obtained from MEDLINE. First, we demonstrate how the support information from MEDLINE can be used to discriminate obvious rules from unexpected ones. This approach is based on the premise that obvious rules are expected to have higher support or higher confidence in the MEDLINE database than unexpected ones. Second, we present an approach for combining rule confidence information from EMR database with that from the MEDLINE database using the minimum discriminative information (MDI) principle [24]. The MDI principle, which is a variant of the maximum entropy (ME) principle but applicable to more general probability distributions, facilitates the probabilistic learning of a calibration function that interpolates the rule confidence obtained from different sources. We demonstrate that the MDI principle not only provides a principled way for calibrating the rule confidence, it can also be tuned according to the reliability of each information source (EMR and MEDLINE).

The remainder of this paper is organized as follows. Section 2 presents related work in association rule mining and use of background knowledge in data mining. Section 3 describes the MDI principle. Section 4 presents our proposed methodology for incorporating background knowledge from MEDLINE database into association rule evaluation. Section 5 presents the experimental results and Section 6 concludes the paper.

2. RELATED WORK

Ever since the problem was first formulated by Agrawal et al. [1], numerous algorithms have been developed for the efficient mining of association rules from large databases [1]. The scalability and efficiency of current state-of-the-art algorithms have enabled users to apply association rule mining at very low support thresholds to discover interesting rules. The drawback, however, is the large number of extracted rules that must be further validated and interpreted by analysts.

Considerable efforts have also been devoted into the development of objective and subjective interestingness measures for association rules. A survey of objective interestingness measures are given in Hilderman et al. [5] and Tan et al. [17]. Statistical issues on rule evaluation (incuding Simpson's paradox and multiple comparison tests) were investigated by Megiddo and Srikant [10], Webb [20] and Freitas [4]. Silberschatz and Tuzhilin [16] proposed a framework for analyzing the subjective interestingness of association patterns based on how unexpected and actionable are the discovered rules. Their approach however did not consider issues such as how to acquire the relevant domain knowledge for rule evaluation. Jaroszewisz and Simovici [6] have applied the Maximum Entropy approach to prune redundant association rules, in which a rule is considered redundant if its confidence is close to the confidence for one of its subrules. Unlike our work, they did not consider the issue of incorporating background knowledge from external sources to improve rule assessment and interpretation.

Previous studies have also considered the use of background knowledge to enhance the performance of other data mining algorithms such as clustering and classification. For example, in [18] Tiffin et al. attempted to find disease causing genes by integrating text mining on MEDLINE database with data mining using the EVOC ontology. The background knowledge encoded in gene ontology is then used to improve the quality of clustering results. In [8] Liu et al. considered the problem of clustering genes with similar expression profiles. They evaluated their clusters by mapping them to the gene ontology categories and comparing the results to that produced by a hypergeometric probability distribution to measure whether a cluster enriched with genes from a particular category is significantly better than that expected by random chance.

The MDI principle and its special case, the maximum entropy principle, have been widely used in the information retrieval and natural language processing literature. Pietra et al. [14] have applied the MDI principle to estimate an n-gram language model that satisfies a set of constraints derived from the training documents. Ratnaparkhi [15] have employed the maximum entropy principle to the part-of-speech tagging problem. On the other hand, Mannila et al. [9] have applied the maximum entropy principle to synthesize global models from frequent itemsets and sequential patterns for query selectivity and protein sequence modeling applications.

Our work builds upon earlier work by Zhu and Rosenfeld [22], who have applied the MDI principle to estimate trigram language models using frequency information obtained from the World Wide Web. Unlike their work, however, we have used the MDI principle as a means for calibrating the rule confidence in order to distinguish obvious from unexpected association rules.

3. PRELIMINARIES

Consider a database $D = \{t_1, t_2, \ldots, t_N\}$ consisting of N binary records (transactions). Let $I = \{i_1, i_2, \ldots, i_d\}$ denote a set of d items from which the transactions are formed, i.e., $\forall i$: $t_i \subseteq I$.

3.1 Itemsets and Association Rules

In association rule terminology, any non-empty subset of items is called an *itemset*. An itemset X is associated with a *support count,* which is defined as: $g(X) = |\{t_i \mid X \subseteq t_i \forall t_i \in D|$. The larger its support count, the more frequent the itemset X appears in the database. The *support* of an itemset X, denoted as s_X, is defined as the fraction of transactions in D that contain X:

$$s_X = \frac{g_X}{|D|} \qquad (1)$$

An itemset is said to be *frequent* if its support is greater than or equal to a minimum support threshold (*minsup*).

An association rule is an implication expression of the form $X \to Y$, where X and Y are itemsets. The support of the rule, denoted as $s_{X\to Y}$, is given by the support of the corresponding itemset $X \cup Y$, i.e., $s_{X \to Y} = s_{X \cup Y}$ The confidence of the rule, denoted as $c_{X\to Y}$, determines how likely items in the set Y appear in transactions that contain all the items in X. In other words,

$$c_{X \to y} = \frac{s_{X \cup Y}}{s_X} \qquad (2)$$

Formally, confidence is the maximum likelihood estimate for the conditional probability P(Y | X) while support is the maximum likelihood estimate for the join probability.

As an example, given an EMR database, I may correspond to the set of diseases, symptoms, and patient complaints while the transactions may refer to each patient's visit to the healthcare provider. Table 1 illustrates three examples of association rules extracted from our EMR database.

Table 1: Sample association rules from EMR database

{pneumonia, pain}→ {pyelonephritis}
{ear pain, nasal congestion} → {cough}
{proteinuria, renal disease} → { hypertension}

In principle, the number of association rules extracted from a given database can be very large. Since not every rule is interesting, the objective of association rule mining is to automatically discover rules that have support ≥ *minsup* and confidence ≥ *minconf*. However, as noted in the Introduction, rules with high support and high confidence might not be subjectively interesting because they are obvious to the domain experts. Domain information is therefore needed to distinguish obvious rules from unexpected ones.

3.2 Minimum Discriminative Information

In this paper, we apply the minimum discriminative information (MDI) principle to combine probability information obtained from the EMR database with that from the MEDLINE database of biomedical citations. To illustrate the idea, let *P and* P_t be a pair of probability models defined on some event space *R* (which might correspond to the set of all possible rules).

Our goal is to learn a probability model *P(r)* that fits the training model $P_t(r)$. Mathematically, the problem is equivalent to finding a model that minimizes the Kullback-Leibler divergence measure defined below:

$$D(p, p_t) = \sum_r p(r) \log(\frac{p(r)}{p_t(r)}) \qquad (3)$$

Note that, in the special case, when P_t is a uniform distribution, minimizing Equation (3) is equivalent to maximizing the entropy of *P(r)*. In the MDI approach, the probability model *P(r)* must also satisfy the following set of equality constraints:

$$\sum_{r \in R} p(r) f_i(r) = s_i \qquad (4)$$

where s_i is a constraint specified on the expected value of a feature f_i defined on the event space *R*. It can be shown that the probability model that minimizes Equation (2) subject to the equality constraints in (4) has the following exponential form:

$$P^*(r) = \frac{1}{Z(r)} p_t(r) \exp(\sum_i \lambda_i f_i(r)) \qquad (5)$$

where λ_is are the Lagrange multipliers of the constrained optimization problem and can be interpreted as the weights of each feature f_i. We will explain how the MDI principle is used to calibrate rule confidence obtained from the EMR data and the MEDLINE database in the next section.

4. METHODOLOGY

This section presents the overall methodology of our proposed approach. We first briefly explain the procedure for generating association rules from the EMR database. We then describe the procedure for incorporating background information from the MEDLINE database into the evaluation and interpretation of association rules.

4.1 Association Rule Generation

Association rule mining algorithms are designed to automatically discover rules that pass the minimum support and minimum confidence thresholds specified by the users. To do this, the following two-step procedure is often employed by most of the current algorithms:

1. ***Frequent itemset generation***. The objective of this step is to automatically extract itemsets that successfully pass the *minsup* threshold (i.e., the frequent itemsets).
2. ***Rule generation***. The objective of this step is to automatically generate rules with confidence ≥ *minconf* from the frequent itemsets.

For the first step, we extracted the frequent itemsets using the PG-Miner algorithm [11]. PG-Miner uses a prefix graph structure to efficiently compress the transaction database and then employ additional pruning strategies from network flow theory to reduce the computational complexity of frequent itemset generation. Once the itemsets have been generated, rules are extracted by partitioning each itemset into two parts—one that forms the left hand side of the rule, while the remaining items form the right hand side.

In principle there are $2^k - 2$ possible rules that can be generated from a frequent itemset of length *k*. For efficient computation, the following rule pruning strategy is used. Given an itemset *X*, let Y_1 and Y_2 be subsets of *X*. If the confidence of the rule $Y_1 \rightarrow X - Y_1$ is less than the *minconf* threshold, then the confidence of the rule $Y_2 \rightarrow X - Y_2$ is guaranteed to fail the threshold whenever $Y_2 \subset Y_1$. For example, if the rule {unary, nocturia } → {kidney} has low confidence, then we can ignore the rule {unary} → {kidney, nocturia} since its confidence is also guaranteed to be low.

4.2 Incorporating Background Knowledge into Rule Evaluation and Interpretation

Let $R = \{r_1, r_2, \ldots, r_k\}$ be a collection of *k* association rules generated from the EMR database *D*, where each rule $r_i \in R$ satisfies the *minsup* and *minconf* thresholds. Given rule $X \rightarrow y$, let $(s_{X \rightarrow y}, c_{X \rightarrow y})$ denote its support and confidence computed from the EMR data whereas $(\hat{s}_{X \rightarrow y}, \hat{c}_{X \rightarrow y})$ are the corresponding support and confidence obtained from the MEDLINE database. In this section, we present two approaches for utilizing the information from MEDLINE in the evaluation of association rules.

4.1.1 Thresholding Approach

Our first approach is based on the following assumption regarding the support (or confidence) of a rule obtained from an authoritative source such as MEDLINE—that the higher the support (or confidence) is, the more obvious the rule is. For example, since the relationship between nasal congestion and cough is fairly obvious, we expect their joint or conditional probability estimate to be relatively high in the MEDLINE literature. On the other hand, unexpected rules (either nonobvious or spurious) are expected to have considerably lower support (and confidence) in the literature.

As a result, we may adopt the following thresholding strategy to distinguish between obvious and unexpected rules using the support (or confidence) values obtained from MEDLINE. Given a rule r: $X \rightarrow y$, if $\hat{s}_{X\rightarrow y} > \tau_s$ (or $\hat{c}_{X\rightarrow y} > \tau_c$), then the rule is considered obvious. Otherwise the rule is classified as unexpected. In practice, the thresholds τ_s and τ_c can either be determined by the domain experts or by examining the distribution of support and confidence values obtained from MEDLINE (e.g., using mean or median). For our experiments, we investigated with all possible thresholds and plot the resulting ROC (receiver operating characteristic) curve by comparing the judgments provided by our domain expert against the rule classification produced by the thresholding approach.

4.2.1 MDI Principle

It is often useful to treat the EMR and MEDLINE databases as two alternative information sources for evaluating association rules. In this situation, we need to develop an approach for combining the rule statistics information from both sources. Given an association rule $X \rightarrow Y$, let $P_t(Y \mid X)$ be the confidence of the rule obtained from the EMR data whereas $P_w(Y \mid X)$ be the confidence obtained from MEDLINE. Our objective here is to learn a calibration function $P^*(Y \mid X)$ that interpolates the confidence values obtained from both information sources. To accomplish this, we will employ the Minimum Discriminative Information (MDI) principle described in Section 3.

4.1.1.1 Formulation

In Section 3.1, we describe the exponential model obtained when minimizing the Kullback-Liebler divergence between two probability models, subject to a given set of linear equality constraints. We now explain how the approach can be used for combining confidence estimates from two sources of information: the EMR and the MEDLINE databases. For brevity, we consider only rules that contain only one item on its right-hand side, i.e., rules of the form $X \rightarrow \{y\}$, where $y \in I$. To further simplify the notation, we denote the 1-itemset $\{y\}$ as y in the remainder of this paper.

Let $R = \{X \rightarrow y \mid s_{X\rightarrow y} \geq \tau$ and $c_{X\rightarrow y} \geq \tau\}$ be the set of rules extracted from the EMR database D. We denote antecedent(R) to be the set of all itemsets appearing on the left-hand side of the rules, i.e., antecedent(R) = $\{X \mid X \rightarrow y \in R\}$. Furthermore, let $V = \{i \in I \mid s_i \geq minsup\}$ denote the set of all frequent items in the EMR data.

We construct a set of binary features $F = \{f_{U\rightarrow v} \mid U \rightarrow v \in R\}$ based on all the rules extracted in R. The value of each binary feature $f_{U\rightarrow v} \in F$ when applied to a rule r: $X \rightarrow y$ is formally defined as follows:

$$f_{U\rightarrow v}(X, y) = \begin{cases} 1 & \text{if } U = X \text{ and } v = y \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

These binary features are used to construct a database of binary transactions (D'), where each transaction corresponds to a rule in R and the "items" correspond to all the features in F. Since the rules are unique, each of them triggers exactly one feature and each feature is triggered only once.

We are now ready to explain how the MDI principle described in Section 3.1 can be adapted to the confidence calibration problem. Specifically, our goal is to learn an aggregated probability model $P^*(y|X)$ that minimizes the following conditional Kullback Leibler divergence:

$$\begin{aligned} & D(P(y \mid X) \,\|\, P_t(y \mid X)) \\ &= \sum_{X,y} P(X,y) \log \frac{P(y \mid X)}{P_t(y \mid X)} \\ &= \sum_{X,y} P(X) P(y \mid X) \log \frac{P(y \mid X)}{P_t(y \mid X)} \\ &= \sum_{X,y} P(X) P(y \mid X) \log P(y \mid X - P(X) P(y \mid X) \log P_t(y \mid X) \end{aligned} \quad (7)$$

subject to the following constraints:

$$\sum_{X,y} P(X) P(y \mid X) f_{U\rightarrow v}(X, y) = s_{U\rightarrow v} \quad (8)$$

Note that $s_{U\rightarrow v}$ denote the support of the feature in D' (i.e., $s_{U\rightarrow v} = 1/|D'|$). Using the Lagrange multiplier method, the objective function to be minimized can now be written as follows:

$$\sum_{X,y} \left[P(X) P(y \mid X) \log P(y \mid X) - P(y \mid X) \log P_t(y \mid X) \right] - \sum_i \lambda_i \left[\sum_{X,y} P(X) P(y \mid X) f_i(X, y) - s_i \right] \quad (9)$$

where we have used the subscript i to denote all the features in F. Taking the derivative of Equation (9) with respect to $P(y|X)$ yields the following conditional probability model:

$$P^*(y \mid X) = \frac{P_t(y \mid X)}{Z(X)} \exp\left[\sum_i \lambda_i f_i(X, y) \right] \quad (10)$$

where $Z(X)$ is a normalization factor and the λ's are the model parameters. Note the equivalence between Equations (5) and (10). To compute Z(X), we need to obtain the confidence estimate from EMR data, $P_t(y|X)$, for all possible rules $X \rightarrow y$, where $X \in$ antecedent(R) and $y \in V$. In turn, this requires us to have the support for all the rules $X \rightarrow y$, including the ones that may be infrequent. Therefore, in order to address this problem, we approximate the confidence $P_t(y|X)$ by $minsup/s_X$ whenever $X \rightarrow y$ is infrequent.

4.1.1.2 Parameter Estimation

Let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_d\}$ denote the set of parameters to be estimated. Our objective is to learn the parameter set Λ using information obtained from the MEDLINE database. Given a rule r: $X \rightarrow y$, let $\hat{g}_{X\rightarrow y}$ and $\hat{c}_{X\rightarrow y}$ be the corresponding rule support count and confidence, respectively, obtained from the MEDLINE database.

Following the approach used by Zhu and Rosenfeld [22], the likelihood function of the rule set with respect to the MEDLINE support counts can be written as follows:

$$L(\Lambda) = \prod_{(X\rightarrow y)\in R} P^*(y \mid X)^{\hat{g}_{X\rightarrow y}} \quad (11)$$

Therefore, the log likelihood function to be maximized can be written as follows:

$$\sum_{(X,y)\in R} \hat{g}_{X\to y}\left[\log P_t(y\,|\,X) + \sum_{U\to v}\lambda_{U\to v} f_{U\to v}(X,y) - \log Z(X)\right] \quad (12)$$

where

$$Z(X) = \sum_{w} P_t(w\,|\,X)\exp\left[\sum_j \lambda_j f_j(X,w)\right] \quad (13)$$

Taking the derivative of Equation (12) with respect to $\lambda_{U\to v}$ yields:

$$\sum_{(X,y)\in R}\hat{g}_{X\to y} f_{U\to v}(X,y) - \sum_{(X,y)\in R}\hat{g}_{X\to y}\left[\frac{\sum_w P_t(w\,|\,X) f_{U\to v}(X,w)\exp\left[\sum_j \lambda_j f_j(X,w)\right]}{Z(X)}\right] \quad (14)$$

Based on the binary feature definition given in Equation (6), the first term in Equation (14) reduces to:

$$\sum_{(X\to y)\in R}\hat{g}_{X\to y} f_{U\to v}(X,y) = \hat{g}_{U\to v} \quad (15)$$

while the second term reduces to:

$$\begin{aligned}
&\sum_{(X,y)\in R}\hat{g}_{X\to y}\left[\frac{\sum_w P_t(w\,|\,X) f_{U\to v}(X,w)\exp\left[\sum_j \lambda_j f_j(X,w)\right]}{Z(X)}\right] \\
&= \sum_{(X,y)\in R}\hat{g}_{X\to y}\sum_w f_{U\to v}(X,w)P^*(w\,|\,X) \\
&= \sum_y \hat{g}_{U\to y}\sum_w f_{U\to v}(U,w)P^*(w\,|\,U) \\
&= \sum_y \hat{g}_{U\to y}P^*(v\,|\,U)
\end{aligned} \quad (16)$$

Setting Equation (14) to zero and using the simplified expressions given in Equations (15) and (16) lead to:

$$\hat{g}_{U\to v} = \sum_y \hat{g}_{U\to y}P^*(v\,|\,U) \quad (17)$$

Since the second term on the right hand side no longer depends on y, it can be factored out. Rearranging the preceding equation yields:

$$P^*(v\,|\,u) = \frac{\hat{g}_{U\to v}}{\sum_y \hat{g}_{U\to y}} = \hat{c}_{U\to v} \quad (18)$$

which means that, using the likelihood function given in Equation (11), the estimated confidence of the rule $X \to y$ is exactly the same as the confidence obtained from MEDLINE database, i.e., $P^*(y|\,X) = \hat{c}_{X\to Y} = P_w(y\,|\,X)$. Next, we describe how to modify the likelihood function in Equation (11) so that it takes into account the confidence from both MEDLINE and EMR databases.

Zhu and Rosenfeld [22] have proposed adding a Gaussian prior term, not only to prevent P^* from overfitting the confidence estimates P_w from the MEDLINE database, but also to provide a systematic way for P^* to also incorporate the confidence values in P_t. Using the maximum a posteriori (MAP) approach, the likelihood function in (11) is modified to:

$$L(\Lambda) = \prod_{X\to y\in R} P^*(y\,|\,X)^{\hat{g}_{X\to y}} \times P(\Lambda) \quad (19)$$

where the parameters in Λ are assumed to have a Gaussian prior with zero mean and the same variance σ^2:

$$P(\Lambda) = \prod_{U\to v}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{\lambda_{U\to v}^2}{2\sigma^2}\right] \quad (20)$$

This leads to the following objective function to be optimized:

$$\begin{aligned}
&\sum_{(X\to y)\in R}\hat{g}_{X\to y}\left[\log P_t(y\,|\,X) + \sum_{U\to v}\lambda_{U\to v} f_{U\to v}(X,y) - \log(Z(X))\right] \\
&- \sum_{U\to v}\left[\frac{1}{2}\log(2\pi\sigma^2) + \frac{\lambda_{U\to v}^2}{2\sigma^2}\right]
\end{aligned} \quad (21)$$

We may use the bound maximization approach [2], which is a variant of the improved iterative scaling algorithm developed by Berger and Della Pietra et al. [2], to estimate the model parameters that maximize the objective function in (21). The idea behind this approach is to incrementally learn the update vector Δ in such a way that guarantees improvement in the likelihood function, i.e., $L(\Lambda+\Delta) \geq L(\Lambda)$. It can be shown that the solution to the bound maximization problem can be written in the following way:

$$\begin{aligned}
&\sum_{(X\to y)\in R}\hat{g}_{X\to y} - \sum_{(X\to y)\in R}\hat{g}_{X\to y}P^*(v\,|\,U)\exp\left[\delta_{U\to v}\right] \\
&- \frac{\lambda_{U\to v} + \delta_{U\to v}}{\sigma^2} = 0
\end{aligned} \quad (22)$$

which yields the update formula of δ. Equation (22) can be solved using Newton's method or any numerical optimization algorithms.

Table 2 summarizes the key steps of our proposed methodology.

Table 2. Proposed Methodology

```
Input: Rule set R = {$X \to y$}, with confidence $P_t(y\,|\,X) = c_{x\to y}$
Output: Set of parameters, $\Lambda = \{\lambda_{U1\to v1}, \lambda_{U2\to v2}, \ldots\}$

Procedure:
1. Initialize the parameter set $\Lambda$.
2. For each rule $X \to y$,
   a. Retrieve support count from MEDLINE, $\hat{g}_{X\to y}$
   b. $\forall (X\to y)\in R$: compute $P^*(y|X)$ using $\Lambda$, $c_{X\to y}$, and $\hat{g}_{X\to y}$.
3. Repeat
   a. Compute $\delta_{X\to y}$ by solving Equation (22)
   b. Update $\lambda_{X\to y} \leftarrow \lambda_{X\to y} + \delta_{X\to y}$
   c. $\forall (X\to y)\in R$: compute $P^*(y|X)$ using $\Lambda$, $c_{x\to Y}$, and $\hat{g}_{x\to Y}$.
   until convergence
```

5. EXPERIMENTAL EVALUATION

This section demonstrates the experimental results we have obtained when applying our methodology to the MQIC (Medical Quality Improvement Consortium) EMR database. The data has been deidentified by the GE Healthcare Data Consortium. The original database contains 5,939,734 clinical visit records of 427,214 patients during the period of 2001-2005. After preprocessing, i.e. removing visit records that contain only items corresponding to medication refills, the remaining number of transactions is 3,115,630. As previously noted, the background information was acquired using the PubMed search interface developed by the National Library of Medicine.

We first extracted the association rules with support more than .00015% and confidence more than 65% using PGMiner and Apriori rule generation algorithm. For brevity, we concentrate only on rules with a single item on the right-hand side. There were 113 such rules that pass the support and confidence requirements in the training data. For evaluation purposes, we have asked our domain expert to classify the rules according to their expectations. Specifically, a rule is classified into two categories: obvious or unexpected (previously unknown or potentially spurious). Based on the classification provided by the expert, we then evaluate the different approaches based on their accuracy and ROC (receiver operating characteristic) curve.

5.1 Evaluation using Thresholding Approach

This section presents our experimental results using the support and confidence thresholding approaches described in Section 4.2.1.

5.1.1 Thresholding on Support

For each association rule $X \rightarrow y$, we query the Pubmed Web service to find the number of abstracts that contain the concepts X and y. This frequency corresponds to the support count $\hat{g}_i$ described in Section 4.2.2.2 We consider such support count as the expected support of the rule.

We argue that if the expected support is reasonably high, (higher than a specified threshold τ) then the extracted rule should be obvious because the relationship between X and y is quite prevalent in the literature. On the other hand, if the expected support of the rule is less than τ, the rule is unexpected because X and y are rarely observed together in literature.

In our experiment, we vary the threshold from the minimum value of $\hat{g}_i$ to its maximum value. At each threshold, we compute its true positive rate and false positive rate by comparing the model's predictions against the classification provided by the domain expert. Figure 1 illustrates the ROC curve for using support from MEDLINE database in comparison to the support from EMR database. As can be seen, the area under ROC curve for MEDLINE database is significantly higher than that from the EMR database. This suggests that it would be useful to augment the background knowledge from MEDLINE database to improve rule assessment.

To further illustrate the predictions made by using support from MEDLINE database, Table 3 illustrates the confusion matrix obtained when τ is chosen to be the median value of the expected support for all the rules. Since the median value was 13, any rule that has an expected support greater than or equals to 13 is considered obvious, while the rest are declared as unexpected rules. Notice that this method could distinguish 47 of 57 rules classified as obvious by our domain expert. The accuracy of this approach is 74.3%.

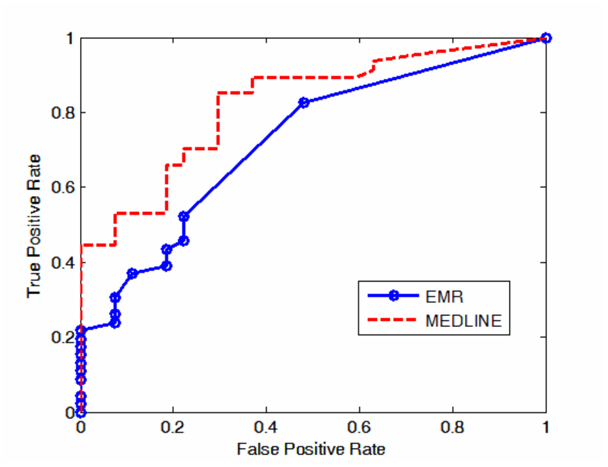

Figure 1: ROC curve for support thresholding.

Swelling $\rightarrow$ leg edema is an example of an obvious rule supported in the literature, where the support in MEDLINE database is 1506, which is considerably higher than the threshold τ. Examples of other obvious rules classified correctly include {obesity, hyperglycemia} $\rightarrow${hypercholesterolemia}, {hematuria, urinary incontinence} $\rightarrow$ {nocturia}.

An example of an association rule classified as unexpected both by the domain expert and the support thresholding approach is {obesity, Grave's disease} $\rightarrow$ {hypertension}. The support count of the rule in MEDLINE database is 2, which is substantially less than the threshold ($\tau = 13$).

Notice that there were 19 rules declared as obvious by the thresholding approach but marked as unexpected. One possible explanation for the false alarm is that the expected support computed from MEDLINE may not be indicative of causal relations between two concepts.

Table 3: Confusion Matrix for pruning rules based on their support in MEDLINE database with $\tau = 13$.

		Predicted	
		Obvious	**Unexpected**
Actual	**Obvious**	47	10
	Unexpected	19	37

For example, the rule osteomalacia ➔ osteomyelitis is classified as unexpected by the domain expert because there is no known causal relation between the two diseases. However, since they are both diseases related to the bone (osteomyelitis is an acute bone infection while osteomalacia is the softening of the bones caused by the deficiency of vitamin D), they often appear together in the literature as examples of possible diagnosis for a bone-related medical condition.

In addition, ten of the rules were declared as unexpected based on the threshold τ but were marked as obvious because some relationships are so obvious that they were hardly mentioned together in the literature (e.g., urinary frequency slowing $\rightarrow$ nocturia).

5.1.2 Thresholding on Confidence

In this experiment we compare the confidence values obtained from MEDLINE against those obtained from the EMR database. The results reported here are for rules with two items on its left hand side (73 rules in total). Given an association rule $X \rightarrow y$, we perform a Web query on PubMed to retrieve the frequency information about the number of abstracts containing X as well as those containing both X and y. We then vary the confidence threshold from the minimum value observed for P_w to its maximum value and plot its ROC curve. The result is shown in Figure 2. For comparison purposes, we have also plotted the corresponding ROC curve for P_t, which is the confidence value computed from the EMR database.

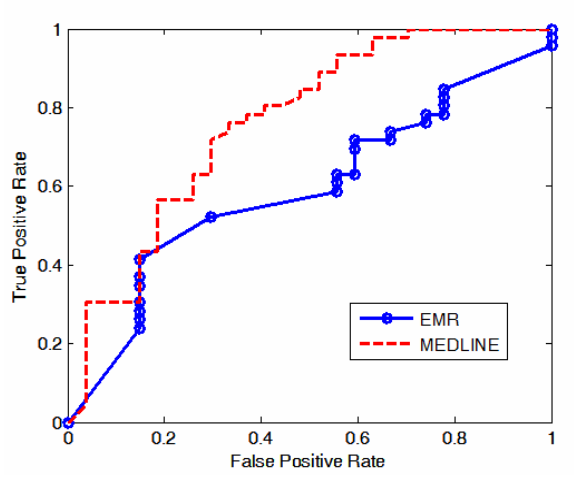

Figure 2: ROC curve for confidence thresholding.

Tables 4 and 5 show the confusion matrices obtained when using the mean value of the conditional probabilities as our threshold for obvious rules. As can be seen, the accuracy for P_w (60%) is higher than that for P_t (59%), which agrees with the previous observation for thresholding based on support.

Table 4: Confusion matrix for P_w

		Predicted	
		Obvious	**Unexpected**
Actual	**Obvious**	22	24
	Unexpected	5	22

When comparing the results of both models (P_t and P_w) we observed that 13 of the obvious rules found by P_t were misclassified as unexpected by P_w. Similarly, 11 of the rules classified by P_w as obvious were misclassified as invalid by P_t. The result is summarized in Table 6.

Table 5: Confusion matrix for P_t

		Predicted	
		Obvious	**Unexpected**
Actual	**Obvious**	24	22
	Unexpected	8	19

This result suggests that it may be useful to combine the classifications made by P_t with those made by P_w. We examine the probability combination approach using MDI principle in the next section.

Table 6: Comparison between predictions by P_w and P_t

	P_t=0 P_w =1	P_t =1 P_w =0	P_t =1 P_w =1	P_t =0 P_w =0
Actual=1	11	13	11	11
Actual=0	1	4	4	18

5.3 Combining Probabilities using the MDI Principle

This section describes the results of our experiments using the MDI principle.

Figure 3 shows the probabilities obtained for the 3 models (P_t, P_w, and P^*) for all 73 rules of length 3 extracted by our association rule mining algorithm. The top curve corresponds to confidence values for P_t while the curve at the bottom corresponds to the confidence values for P_w. This suggest that MEDLINE literature tends to underestimate the magnitude of the confidence, but still, as shown in the previous section, it provides a much better indicator for obvious rules than P_t.

In principle, depending on the hyperparameter σ, P^* can be tuned to either fit more closely to P_t or P_w. Figure 4 shows how the ROC curve for P^* changes for $\sigma = 0.0001$, 1, and 100. Notice that the curve approaches P_t when σ goes to zero. On the other hand, in the asymptotic limit when σ goes to infinity, the ROC curve gets closer to P_w. These limits can be theoretically verified by referring back to Equations (10) and (22).

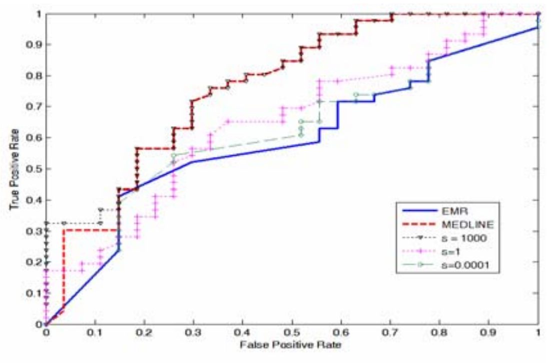

Figure 4: ROC curves for different values of σ.

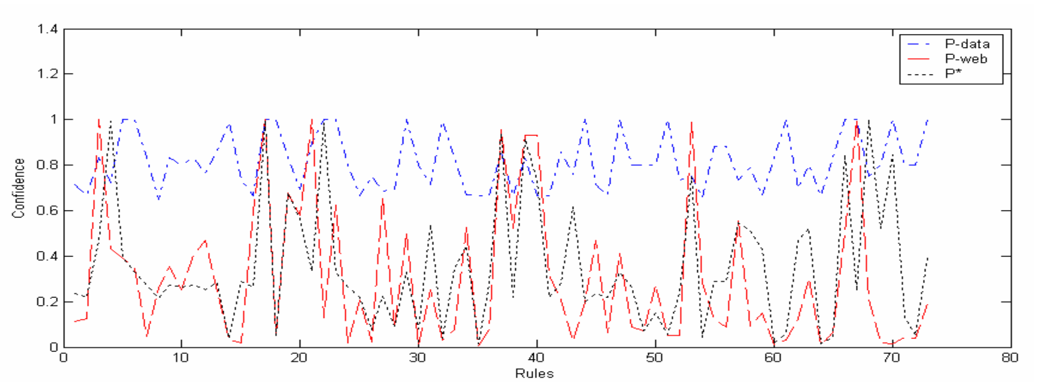

Figure 3: P*, P_t, P_w curves

Finally, we show the confusion matrix obtained when using the mean value of *P** as the threshold. We obtained an accuracy of 66% and a precision of 81% and recall of 62 %.

Table 7: Confusion Matrix for *P*

		Predicted	
		Obvious	**Unexpected**
Actual	**Obvious**	29	17
	Unexpected	7	20

We further compared the rules that were predicted to be valid by P* and missed by both P_t and P_w. For example we found that 6 rules were captured and correctly predicted to be valid by *P** (TP) and missed by P_t and P_w , while 5 rules were missed by only P_t and 2 rules were missed by P_w but captured by P* and P_t. Again this implies that that the interpolated model could be a good choice between the two extremes.

6. CONCLUSION

Evaluating the subjective interestingness of association rules is a challenging task due to the difficulty in acquiring and incorporating the background information needed to determine whether a rule is obvious or unexpected. This paper presents a principled way for utilizing background information acquired from an authoritative source on the World Wide Web to evaluate the quality of association rules. We present two approaches for utilizing the rule statistics information from the authoritative source. The first approach detects the obvious rules using a support or confidence thresholding strategy. The second approach employs the Minimum Discriminant Information (MDI) principle to combine the rule confidence estimates obtained from the two different sources. The MDI principle allows us to calibrate the confidence estimate according to the reliability of the information sources.

For future work, we will extend the methodology to handle confidence estimates from more than two information sources. We also plan to investigate other types of constraints that can be incorporated into the MDI framework. For example, the equality constraints in the MDI framework may seem too strict for some domains. Therefore, it would be useful to modify the formulation to handle inequality constraints.

7. ACKNOWLEDGMENTS

The authors would like to thank Dr. Henry Barry, Dr. Michael Zaroukian, and Dr. David Weismantel from the College of Human Medicine at Michigan State University for providing them with the EMR data and the domain expertise.

8. REFERENCES

[1] Agrawal R., Imieliński T. and Swami A. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Record,* 22(2), 207 – 216, 1993.

[2] Berger A.L., Della Pietra V.J. and Della Pietra S.A. ,Maximum Entropy Approach for Natural Language Processing. *In Proceedings of Computational Linguistics.* 22(1), 39 – 71, 1996.

[3] Chen S. F. and Rosenfeld R. A Gaussian prior for smoothing maximum entropy models. In *Proceedings of CMU-CS-99-108,* 1999.

[4] Freitas A.A. On Objective Measures of Rule *Mining and* Surprisingness, In *Proceedings of Principles of Data Knowledge Discovery*, 1-9, 1998.

[5] Hilderman R.J. and Hamilton H.J. Evaluation of Interestingness Measures for Ranking Discovered Knowledge. In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining,* 247-259, 2001

[6] Jaroszewicz S. and Simovici D.A. Pruning Redundant Association Rules Using Maximum Entropy Principle, In *Proceedings of Advances in Knowledge Discovery and Data Mining, 6th Pacific-Asia Conference,* (2), 135-147, 2002

[7] Li W., Han J. and Pei J. CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In *Proceedings of IEEE-ICDM,* 369-376, 2001

[8] Liu J., Wang W. and Yang J. Gene ontology friendly biclustering of expression profiles. In *Proceedings of Computational Systems Bioinformatics Conference of IEEE*, 436-447, 2004.

[9] Mannila H., Pavlov D. and Smyth P. Prediction with Local Patterns using Cross-Entropy. *In Proceedings of the fifth ACM SIGKDD,* 357 – 361, ISBN:1-58113-143-7, 1999.

[10] Megiddo N. and Srikant R. Discovering predictive association rules. In *the Proceedings of the ACM Int'l Conf. on Knowledge Discovery and Data Mining,* 274 -278, 1998.

[11] Moonesinghe K., Fodeh S.J. and Tan P.N. Frequent closed itemset mining using prefix graphs with efficient flow-based pruning strategy. In *Proceedings of the Sixth ICDM*, 426 – 435, 2006.

[12] Nigam K., Lafferty and Maccallum J. Using Maximum Entropy for Text Classification. In *Proceedings of IJCAI-99 Workshop on Machine Learning for Information*, 1999.

[13] Ozgur A, Tan P.N. and Kumar V. RBA: An Integrated Framework for Regression Based on Association Rules. In *Proceedings of SIAM Intl. Conf. on Data Mining,* 2004

[14] Pietra S.D., Pietra V.D., Mercer R.L. and Roukos S. Adaptive language modeling using minimum discriminant estimation, In *Proceedings of Workshop on Speech and Language*, 103-106, 1992

[15] Ratnaparkhi A. A Maximum Entropy Model for Part-Of-Speech Tagging, In *Proceedings of Conf. on Empirical Methods in Natural Language Processing,* 133-142, 1996

[16] Silberschatz A and Tuzhilin A. What Makes Pattern Interesting in Knowledge Discovery Systems. In *Proceedings of IEEE Trans. Knowl. Data Eng.,* 8 (6), 970 – 974, 1996.

[17] Tan P.N., Kumar V. and Srivastava J. Selecting the right objective measure for association analysis. . In *Proceedings of Data Mining and Knowledge Discovery,* 29(4), 293 - 313, 2004.

[18] Tiffin N., Kelso J. F., Powell A. R., Pan H., Bajic V. B. and Hide W. A. Integration of text-and data-mining using ontologies successfully selects disease gene candidates. *Nucleic Acids Research,* 33(5), 1544 -1552, 2005.

[19] Wang C. and Parthasarathy S. Summarizing itemset patterns using probabilistic models. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining,* 730-735, 2006

[20] Webb G. Discovering significant rules. In *Proceedings of the ACM Int'l Conf. on Knowledge Discovery and Data Mining*, 434 – 443, 2006.

[21] Zhang S. and Zaki M.J. Mining Multiple Data Sources: Local Pattern Analysis. In *Proceedings of Data Mining and Knowledge Discovery*, 121-125, 2006.

[22] Zhu X. and Rosenfeld R. Improving trigram language modeling with the world wide web. In *Proceedings of ICASSP*, 533–536, 2001.

[23] http://www.ncbi.nlm.nih.gov/entrez/query.fcgi

[24] http://mathworld.wolfram.com/RelativeEntropy.html

Mining Vector-Item Patterns for Annotating Protein Domains

Jianfei Wu and Anne M. Denton
Department of Computer Science
North Dakota State University
Fargo, ND
{jianfei.wu,anne.denton}@ndsu.edu

ABSTRACT

The diversity of information collected in bioinformatics and other application areas makes it increasingly important to develop techniques for associating data of different types. An algorithm is introduced for finding patterns involving item and vector data. The presence of an item defines a subset of vectors that may or may not show unexpected density fluctuations. We test for fluctuations by studying density histograms. Vector-item patterns are considered significant if their density histogram significantly differs from what is expected for random subsets of transactions. We evaluate our algorithm on yeast cell cycle gene expression data in combination with domain annotations from the Interpro database and on time series subsequence data.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Pattern Analysis*

General Terms

Algorithms, Experimentation, Performance

Keywords

Pattern mining, Vector data, Pattern significance

1. INTRODUCTION

For many years, disk space has grown exponentially at rates higher than Moore's law for the number of microprocessors on a chip. A consequence is not only that larger tables are stored, but also that the diversity of data associated with any one entity has increased. In bioinformatics the increased storage capacity has been matched with developments in high-throughput experimentation. It is typical that any one protein is not only characterized by its sequence and derived sequence signatures, but also by results from high-throughput experimentation, text from scientific publications related to the protein, and other types of data.

Many algorithms for finding frequent or characteristic patterns have been developed for item data [4, 5, 19], continuous data [10], and combinations thereof [32, 28, 12]. The assumption behind such algorithms is that each of the continuous attributes represents a separate fact that may independently be related to any combination of the categorical attributes. This assumption, however, is not always valid. Multiple continuous attributes often represent a coherent concept that may or may not be related to items in the database. Gene expression experiments may be performed as time course experiments, in which different attributes correspond to the same overall experimental condition and differ only in the time that has passed since the beginning of the experiment. Feature vectors in image analysis and word vectors in text mining, are also examples of continuous data for which similarity is normally determined from a combination of many or all of the attributes.

Figure 1 illustrates the problem. Each of the circles represents an object or transaction. The spatial position of the object corresponds to a vector attribute, that is — in this example — two-dimensional. In general, a vector attribute is composed of D continuous attributes that are assumed to form a vector space. In Figure 1, circles that are solid black represent objects, for which a particular item is present. Only a single item is represented in this image, but the process can be applied to many items. In the left panel the solid black circles are close together. For each of the solid circles all of the other circles can be considered "close" according to the closeness criteria we develop in Section 3.4. The histogram under the left panel reflects the observation that there are five objects that have the item of interest, each of which has six neighbors that have the item. The right panel shows objects with the same vector data as the left panel, but the item data are associated with different objects. Although the vector data are identical, the item data look far more distributed, and the histogram shows that relevant objects only have two to four neighbors. The setting on the left side illustrates what we consider a vector-item pattern.

A natural test as to whether the histograms that represent the data are exceptional is based on random permutations. If a random selection of an equal number data points leads to a similar histogram, then the selection criterion / item is not related to the vector data. We will show in Section 3.5 that the expected distribution of points can also be evaluated analytically.

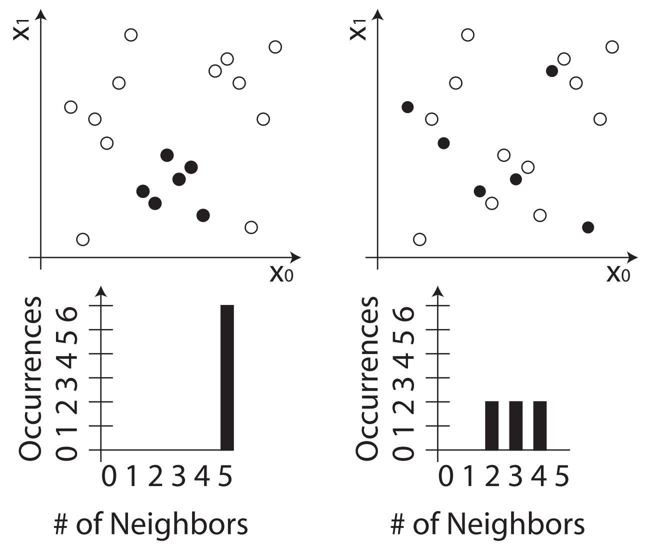

Figure 1: Sketch of a vector-item pattern between a 2-dimensional vector and one item. Records are represented by circles. The vector data determine the positions of the circles in the plane; existence of the item is represented as solid black filling. Bottom: Histograms summarize the distribution of neighbors with the item. Left: Records that have the item are close (vector-item pattern noticeable). Right: Same vector data as left but item data are distributed such that no vector-item pattern is noticeable.

So far, biologists who do time-course experiments are faced with cumbersome gene lists that are difficult to interpret. They may notice that several of the genes that are similarly expressed also have certain interesting domains or functional annotations. Such reasoning corresponds to a two-step process, in which the expression data is first analyzed with traditional techniques and relationships with domain or functional data are studied as a separate step. We use this hypothetical two-step process for comparison purposes in Section 4. Our approach removes the artificial separation of the two steps and directly returns the domain or functional information.

In our algorithm, the vector data are normalized based on rank order. We then use the existence of a categorical attribute value as filter to derive subsets. For each of those subsets we create a density histogram that summarizes the number of neighbors to each point. A point is considered a neighbor if the number of dimensions in which it is within a predefined range exceeds a predefined threshold. We study the choice of range and threshold parameter extensively. We show that our algorithm is both effective and efficient for an example data set consisting of Interpro domain information [14], and gene expression data from cell-cycle experiments [31] for all yeast genes.

2. RELATED WORK

Much work has been done on clustering gene expression data to find the genes that show a similar differential expression pattern under the given conditions [20, 16, 6, 22, 29, 13]. While this work does not directly relate genes to domain or functional information, it is worth noting that functional information is sometimes used to improve clustering results [21, 23]. Simultaneous identification of functional groups and the genes that belong to them has been achieved through biclustering techniques [11]. Such approaches assume that the vector data show multiple groups, which is not assumed in our approach. The significance of clustering results has been studied in the context of validation of clustering approaches [34], and functional information has been considered in this context [8]. Expression patterns have also been used to identify differentially expressed genes in time course experiments [15, 17, 7, 31].

It has frequently been observed that the Euclidean distance and other L_p norms are problematic in high dimensions [2], and new distance measures have been developed [1] that only consider a subset of dimensions over which points are close. We use a similar concept of requiring a fraction of dimensions to be within a given range. We show that such a distance measure can be interpreted as a subspace-based evaluation. In contrast to conventional subspace clustering based on axis-parallel projections [3], our distance measure evaluates similarity in any of the projections. More importantly our algorithm searches for subsets of the data that show inhomogeneities rather than looking for clusters in the full set of points.

The normalization in this paper is related to quantile normalization [9] in that the quantile of a point determines its normalized value. In our normalization we consider a constant distribution as reference distribution. Such a normalization results in distances that are closely related to the mass-distance discussed in [35].

3. ALGORITHM

3.1 Vector and Item Data

We consider D continuous attributes, $x_i \in \mathbb{R}, 0 \leq i < D$, that are known to be related based on background knowledge, as one "vector" attribute $\mathbf{x}$ with domain $\text{dom}(\mathbf{x}) = \mathbb{R}^D$. The set of occurring data points (extant domain) is $V \subset \mathbb{R}^D$. In principle, a data set can have arbitrarily many vector attributes. Each of the vector attributes is considered separately in the pattern mining step. Note that the continuous attributes that form vector attribute $\mathbf{x}$ do not have to come from the same source. Different combinations of attributes can, furthermore, be considered separately. In the evaluation, we determine patterns involving four sets of experiments as separate vector attributes and also search for patterns involving the full set of all continuous attributes.

Similar to the original formalism of Agrawal et al. [4] we consider item data as Boolean attributes $B^{(i)}, 0 \leq i < M$, that represent the presence of item i, with M distinct items occurring in the database. Conjunctions of the Boolean item attributes could be considered and, among item data, the usual downward closure statement based on support remains valid. However, the significance measure of vector-item patterns does not satisfy the downward closure property. For the evaluation in Section 4 we limit our discussion to patterns that involve individual item and vector attributes and, thereby, avoid using a support threshold beyond the requirements of the statistical analysis.

A data set with a single vector attribute is defined through the relation $R(V, B^{(1)} \ldots B^{(M)})$. Generalization to multiple vector attributes is straightforward.

3.2 Outline of the Algorithm

To find patterns among vector and item data we perform the following steps

Normalization Vector Data are normalized such that they homogeneously fill a given interval in each individual dimension. Details of the normalization are given in Section 3.3

Density Histogram Computation For each item, a density histogram analogous to the bottom part of Figure 1 is calculated. Density histograms are at the center of the algorithm and are discussed in Section 3.4.

Computation of Expected Histogram For each item, significance is determined in comparison with the expected histogram. The expected histogram can either be calculated based on theoretical considerations or through random sampling. Both approaches are discussed in Section 3.5.

Determining Significance Once the observed and expected histograms are known, the significance is determined using a χ^2 test. Those domains that are significant at the 5% level are returned as having a strong pattern with the vector data under consideration.

Considering Multiple Vector Data Sets The process can be repeated for different vector data sets.

3.3 Normalization

The basic idea for normalizing attributes such that they are evenly distributed in each dimension is to convert to rank order and scale the result accordingly. One may consider the following definition

$$x'_i = \frac{1}{N} \int_{-\infty}^{x_i - \epsilon} \sum_{j=1}^{N} \delta(x - x_j) \mathrm{d}x \qquad (1)$$

where $\delta(x)$ is the Dirac delta function and $\epsilon > 0$ is a number smaller than all differences between attribute values. Note, however, that this transformation maps equal values into equal values. That means that some attribute values may occur multiple times, and the modeling through a constant distribution is less accurate.

This problem is more serious than it may initially appear: In practice, gene expression data are not available with arbitrary precision, and the same holds for most other types of data that are commonly called "continuous". The data set we used for the evaluation only lists two digits after the decimal point. Considering that the data set has over 7000 records, some values occur more than 100 times. We cannot uniquely decide on a rank order among these, nor do we expect that the experimental precision is high enough to make such a rank order meaningful. However, assigning the same normalized value multiple times results in a poor fit with the model of a constant distribution. In practice, we assign a random ordering to the records. This is not expected to increase the inherent error since experiments are typically designed such that the resolution of the output values is higher than the precision of the experiment.

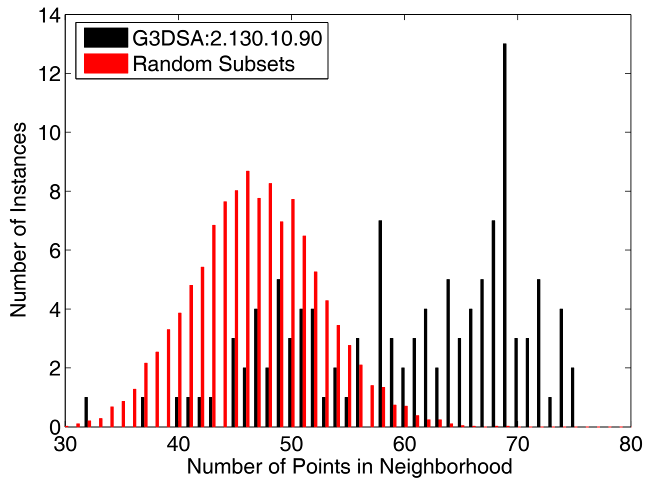

Figure 2: Histograms for a real domain from the Interpro database (black) and a random subset of genes (red) with the same number of elements.

3.4 Density Histograms

The goal of the density histogram computation is to summarize the distribution of data points with respect to the item of interest. Each Boolean attribute $B^{(i)}$ defines a subset of the vector data. For each of these subsets the density is summarized. The sub-relation $R^{(i)}$ defined by item $B^{(i)}$ is given through the selection (σ)

$$R^{(i)} = \sigma_{B^{(i)}}(R) \qquad (2)$$

and the corresponding set of points is given through a projection (π) to vector attribute V

$$V^{(i)} = \pi_V(R^{(i)}). \qquad (3)$$

For each $V^{(i)}$, we summarize the density distribution through a histogram. We assume that a neighbor selector function c has been defined that has the following property

$$c(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{y} \text{ is to be considered a neighbor of } \mathbf{x} \\ 0 & \text{else} \end{cases} \qquad (4)$$

For the density definition of conventional density-based clustering with a uniform kernel, the neighbor selector would be

$$c_{\text{uniform}}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if} \quad |\mathbf{x} - \mathbf{y}| < d/2 \\ 0 & \text{else} \end{cases} \qquad (5)$$

where d is the diameter of the hypersphere that defines a neighborhood. Note that c is of type integer and does not satisfy the normalization conditions of a kernel function.

Given c($\mathbf{x}$,$\mathbf{y}$), a density histogram of V can be defined such that each $\mathbf{x}$, for which the sum of neighbors is equal to k,

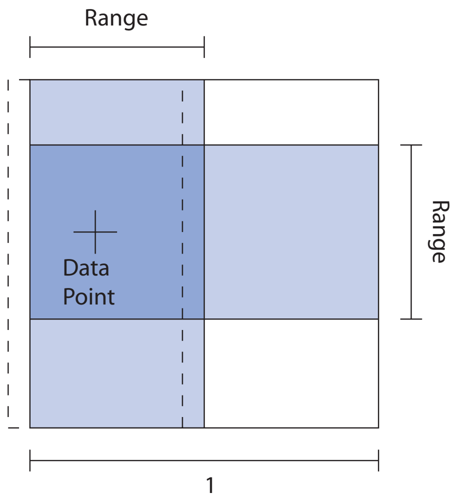

Figure 3: Illustration of subspaces in two dimensions. Subspaces that extend beyond the volume defined by the normalization procedure are shifted to fit completely inside.

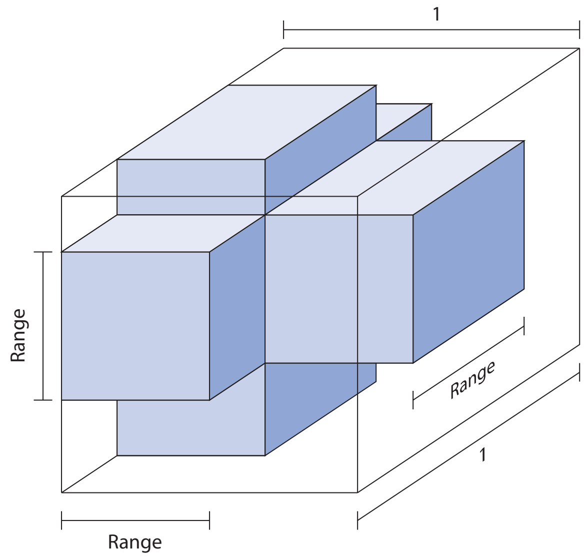

Figure 4: Illustration of the volume that defines the neighborhood of an example point in three dimensions, with range $r = 0.5$. Threshold t could be anywhere in the interval $]1/3, 2/3]$ to yield this image.

contributes 1 to h_k

$$\begin{aligned} h_k &= \sum_{\mathbf{x} \in V} \delta_{k,s(\mathbf{x})} \quad \forall k \\ \text{with} \quad s(\mathbf{x}) &= \sum_{\mathbf{y} \in V} c(\mathbf{x}, \mathbf{y}), \end{aligned} \tag{6}$$

where $\delta_{i,j}$ is the Kronecker delta, which is 1 when $i = j$ and 0 otherwise. Figure 2 shows the histogram for a real domain (black) and for an average over random subsets of transactions (red).

In this paper, we use a neighbor selector for which we only require a fixed fraction of dimensions to match rather than the uniform one of equation (5). We consider two vectors as similar if the number of dimensions in which they are similar exceeds a threshold tD, with D being the total number of dimensions and $t \in]0, 1]$.

$$c_{\text{subspace}}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if} \quad \sum_i c_{1d}(x_i, y_i) \geq tD \\ 0 & \text{else,} \end{cases} \tag{7}$$

where $c_{1d}(x_i, y_i)$ is a one-dimensional neighbor selector. Note that for $t = 1$ this definition requires values in each dimension to satisfy $c_{1d}(x_i, y_i) = 1$. For $c_{1d}(x_i, y_i) = |x_i - y_i| < r/2$ this corresponds to defining distances on the basis of the MAX metric, i.e. the distance is determined by the maximum of distances in individual dimensions. In practice, we found it useful to limit t to $t \in [0.2, 0.8]$.

The neighbor selector of equation (7) can also be interpreted as being subspace-based. Requiring that a point be close in a fraction of t dimensions is equivalent to saying that the point should be within any one of the axis-parallel projections of a hypercube that satisfy the requirements of the one-dimensional selector $c_{1d}(x_i, y_i)$ and the maximum number of dimensions over which projections have been performed $(1 - t)D$.

Figure 3 illustrates the subspaces for two dimensions in which a minimum of 1 dimension is required to match, corresponding to a t in the interval $]0, 0.5]$. In the light shaded areas only one of the dimensions respectively corresponds to regions within the range of the data point $\mathbf{x}$.

It is important for our concept that neighbor distributions of randomly distributed points only depend on the number of points and not on the position of the central point with respect to the normalization volume. For that reason we have to ensure that the volume for which $c_{\text{subspace}}(\mathbf{x}, \mathbf{y}) = 1$ does not depend on $\mathbf{x}$. We do so by shifting this volume such that it does not extend beyond the boundaries of the volume defined by the normalization. The dashed lines in Figure 3 illustrate an area that is defined by $[x_1 - 0.5, x_1 + 0.5]$ in two dimensions. Note how this area exceeds the boundaries of the square with side length 1 over which the normalized data extend. The shaded area illustrates the shifted area, which is entirely enclosed in the square. In general, the one-dimensional neighborhood selector $c_{1\text{d}}(x_i, y_i)$ is defined as

$$c_{1\text{d}}(x_i, y_i) = \begin{cases} 1 & \text{if} \quad (y_i > \min(\frac{1}{2} - r, x_i - \frac{r}{2})) \\ & \qquad \wedge (y_i < \max(-\frac{1}{2} + r, x_i + \frac{r}{2})) \\ 0 & \text{else} \end{cases} \tag{8}$$

This definition is not symmetric in $\mathbf{x}$ and $\mathbf{y}$, i.e. point $\mathbf{y}$ may be within range of point $\mathbf{x}$ without point $\mathbf{x}$ being within range of point $\mathbf{y}$. We can use such a measure since we are only interested in the statistical properties of sets of points, and do not make statements about individual points. Figure 4 shows the subspace setup in 3 dimensions. An example volume with $r = 0.5$ and $t = 0.5$ is highlighted. In three dimensions, $t = 0.5$ means that a data point must be within the specified range for at least 2 dimensions.

Algorithm 1: Density Histogram Algorithm

```
Data: $pts$;            /* nPts data points */
Data: $items$;
Result: $significance$;   /* of each item */
1  $normPts$ = normalize($pts$);
2  $hist$ = zeros(1,nPts); /* vector of zeros */
3  foreach $it \in items$ do
4      $itemPts$ = findPoints($normPts$,$it$);
5      foreach $x \in itemPts$ do
6          $dens$ = NumberOfNeighbors(x);
7          $hist(dens)$++;
8      $randHist$ = zeros($1, nPts$);
9      for $i$=$1$:$nAv$ do
10         $randPts$ =
           randSubset($normPts$,occurrences($it$));
11         foreach $x \in randPts$ do
12             $dens$ = NumberOfNeighbors($x$);
13             $randHist(dens)$++;
14     $randHist / = nAv$;
15     $significance(it)$ =
       chiSquaredGoodnessOfFit($hist, randHist$);
16 return $significance$
```

3.5 Determining Significance

We search for vector-item patterns by comparing observed histograms with their expected counterparts. Hence, we must first determine the expected distribution for the given number of transactions. In principle, a theoretical derivation can be used for this purpose. We first consider the probability of a data point being within the given range for exactly one axis parallel projection of the D-dimensional space. Let us assume that the projected space is d-dimensional. The probability of a point being within range r of the total interval $[-0.5, 0.5]$ for any one dimension is $p = r$. The probability of being within range r for d dimensions is $p = r^d$. We assume that the point is not within range for the remaining $D - d$ dimensions. The probability of being in precisely the subspace under consideration is

$$p_{\text{one subspace}} = r^d (1-r)^{D-d} \tag{9}$$

The number of subspaces with d dimensions is $D!/d!(D-d)!$. Hence, the probability of finding a point within any subspace of at least $d = \lceil tD$ dimensions, where "$\lceil$" is the ceiling operator, is given by the cumulative binomial probability density function

$$\begin{aligned} p &= \sum_{i=d}^{D} \binom{D}{i} r^i (1-r)^{D-i} \\ &= \text{binocdf}(D-d, D, 1-r) \end{aligned} \tag{10}$$

where Matlab notation was used in the second line.

Given this probability p, we can calculate the expected distribution for the density histogram h. This distribution depends on the number of transactions N_i that are selected by the item attribute of interest b_i:

$$h_k = N_i \binom{N_i}{k} p^k (1-p)^{N_i - k} \tag{11}$$

Although equations (10) and (11) are both governed by underlying binomial distributions, they represent entirely different properties of the system. Equation (10) makes a statement about one point relative to another point. The probability of the point being within the neighborhood of the other is given by a binomial distribution, because the D dimensions of the difference vector are considered to be independent. The point is only required to be within range for d out of a total of D dimensions. Equation (11) makes a statement about all points in one vector subset V_i. We evaluate the neighborhoods for each one of the points as center. For each center, we know that any one other point is in a neighborhood with probability p. The probability that k points out of the total of N_i points are within the neighborhood is given by a binomial distribution. We sample the distribution by picking every point as center once, hence the prefactor N_i.

We compare the observed and expected distribution using a standard χ^2 goodness-of-fit test. Bins at both ends of the distribution are merged until the expected number is at least 5. If intermediate bins have an expected number smaller than 5 then pairs of bins are merged until no more bins have an expected number < 5 (one recommended strategy according to [26]). A vector-item pattern is considered significant if the χ^2 goodness-of-fit test yields a p-Value < 0.05.

In practice, we derive the expected distribution by averaging over a large number of randomly picked subsets (in the evaluation we use 50). This step ensures that the expected distribution appropriately reflects the real distribution even when the distribution of the complete data set deviates from the fully random model. Such deviations can occur since we can only fix the distribution of individual dimensions. Correlations among attributes can potentially still lead to in-

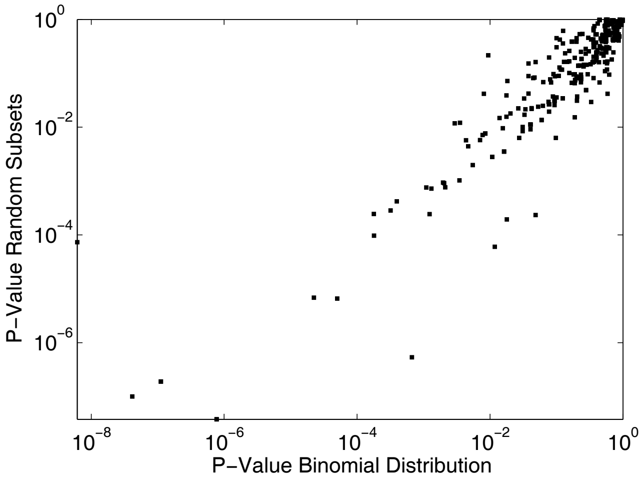

Figure 5: P-Value for all Interpro domains comparing with a binomial distribution (rightward) and an experimentally determined comparison distribution (upward).

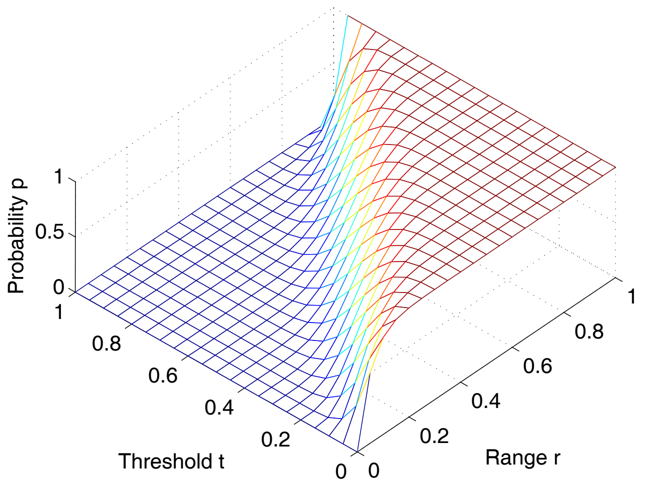

Figure 6: Probability p of a point satisfying r (range) and t (threshold) criteria for D=20.

homogeneous distributions in multi-dimensional subspaces and the full space. The data set we use also suffers from many missing values. In Figure 5 an averaged distribution is plotted against the theoretical binomial distribution. For this plot, imputation was used and missing values were replaced by zeros before normalization. It can be seen that, although the fit is not perfect, p-Values based on both types of evaluation are closely related.

The complete algorithm can be seen as Algorithm 1. An iteration is performed over all items and, for each item, densities are evaluated at the location of each point that has the given item. "findPts" selects those vectors for which the item is present, and "randSubset" selects a random subset of the same number of points. "NumberOfNeighbors" function is given by a sum over equation (7)

$$dens(x) = \sum_{y \in pts} c_{\text{subspace}}(\mathbf{x}, \mathbf{y}) \tag{12}$$

A faster version of the algorithm would use the equation 11 to determine the comparison distribution, but in this paper we opted for accuracy of the result.

3.6 Parameter Choices

Our algorithm allows, in principle, choice of the two parameters, r (range) and t (threshold). Fortunately, one of the parameters can be chosen based on fundamental considerations alone: Our algorithm is expected to be most sensitive to local density fluctuations if the probability that enters into equation (11) itself is sensitive to fluctuations. In the random model with uniform distribution, p depends on input parameters t and r as shown in Figure 6 for $D = 20$. The slope of the distribution is largest for $rD = d$, which can be seen as follows: The maximum of the slope of a cumulative density function is found at the maximum of the corresponding probability density function, which is at $D - d = (1 - r)D$, or $d = rD$ for the distribution in equation (10). Since all our data sets have $D > 10$, we ignore the ceiling operator in $d = \lceil tD$ and choose $r = t$ for all of our experiments, regardless of whether tD is an integer. The choice of the remaining parameter $r = t$ is somewhat data set dependent. We evaluate different choices and find that typically $r = t = 0.5$ leads to the most stable results.

Table 1: Expression Data Sets

Name	Abbr.	No. of Att.	No. of Genes
Alpha	Alp	22	6177
Cdc15	C15	24	5995
Cdc28	C28	17	6147
Elu	Elu	14	6075
—	All	77	6178

4. EXPERIMENTAL EVALUATION

The algorithm is implemented in MATLAB, allowing us to access many of the statistics functions directly as part of the Statistics Toolbox. For simplicity, we use a bitvector representation for items, since vectors can be handled much more easily in MATLAB than sets. We did not encounter memory limitations for the data sets we considered.

4.1 Evaluation on Gene Expression Data

In our first evaluation we use gene expression data sets from cell cycle experiments on yeast [31] as vector data, which are available at [30]. Table 1 summarizes the properties of the four data sets from separate experiments.

Item data come from the Interpro database, in which information on protein domains, motifs, and other kinds of sequence signatures is collected and combined [27]. For simplicity, we refer to all sequence signatures as domains. Yeast domains are available at [14]. We limit our study those domains for which significance can be evaluated without violating the constraint of having at least 5 instances in at least 2 bins, limiting the set to 329 domains.

We evaluate effectiveness by applying the algorithm separately to the four data sets corresponding to different experiments. For each data set, we determine the domains that are significantly related to the vector data at the 5% level. We then compare the two sets using a χ^2-test on the contingency table of the results. We use a χ^2-test with Yates correction for all our tests on contingency tables to account for possibly small cell values [33].

As an example, the results for the Alpha data set and the Cdc15 data set have the following contingency table:

	C15	$\overline{\text{C15}}$	tot
Alp	106	61	167
$\overline{\text{Alp}}$	54	108	162
tot	160	169	329

The contingency table shows that for almost two thirds of the predictions there is agreement between both data sets. This is a reasonably good result, considering, that the experiments were performed independently. The p-value for this contingency table is 4.52E-8, indicating that it would be highly unlikely to get such an overlap accidentally. Table 2 summarizes the mutual overlap and p-values for all data sets we considered, using $r = 0.5$ as range. The com-

Table 2: Results for $r = 0.5$

	Sp	All	Alp	C15	C28	Elu	
	54	41	34	38	40	34	Sp
		219	138	133	146	128	All
All	0.11		167	106	124	100	Alp
Alp	0.05	**5E-10**		160	114	104	C15
C15	**5E-4**	**8E-10**	**5E-8**		193	109	C28
C28	**0.01**	**5E-5**	**7E-9**	**8E-6**		159	Elu
Elu	**0.02**	**3E-7**	**3E-5**	**5E-9**	**5E-4**		

bination of all data sets, resulting in a single data set with 77 columns ("All") is also shown. The diagonal represents the number of domains found as significant for each of the techniques. The upper triangular matrix shows the overlap between any two of the experiments. Set below is a lower triangular matrix that gives the p-value of the contingency table corresponding to the two data sets. P-values that are below the 5% significance level are rendered in bold face.

Table 2 furthermore shows a comparison with a traditional type of analysis. For this comparison we created contingency tables of genes that have a particular domain compared with genes that have been identified as cell cycle genes in [31]. Significance is determined using a χ^2 test with Yates correction. 54 domains are significant in this analysis. Note that the total number of domains that are considered significant is larger for our own evaluation, which identified up to 219 domains as significant.

We now look at an example domain in more detail. The domain G3DSA:2.130.10.90, which was used as an example in Figure 2, occurs 117 times in the yeast data set. Comparing the occurrence of this domain with the Spellman subset, we get the following contingency table:

	G3DSA	$\overline{\text{G3DSA}}$	tot
Spellman	9	790	799
$\overline{\text{Spellman}}$	108	5271	5379
tot	117	6061	6178

The p-value based on χ^2-test with Yates correction is 0.071, which is not considered significant, and the number of occurrences of domain G3DSA:1.25.40.20 in the genes from the Spellman set (9 times) is actually smaller than would be expected by chance (15.1 times). In our own analysis, using the combined data set, this domain has p-value that is below the smallest double precision value in MATLAB and is hence considered significant, which is in accordance with the impression gained from Figure 2.

The overall consistency among our results gives a strong indication that our algorithm is able to extract reliable expression - domain patterns. Table 2 shows that the relationship between results from our algorithm is consistently significant, even when expression data are from independent experiments. Overlap is particularly high for comparisons with the full data set. Naturally, any individual data set is dependent of the full data set. The high overlap in results is noteworthy rather because it shows that our algorithm performs as well or better for 77-dimensional vector data as for the sets with fewer attributes.

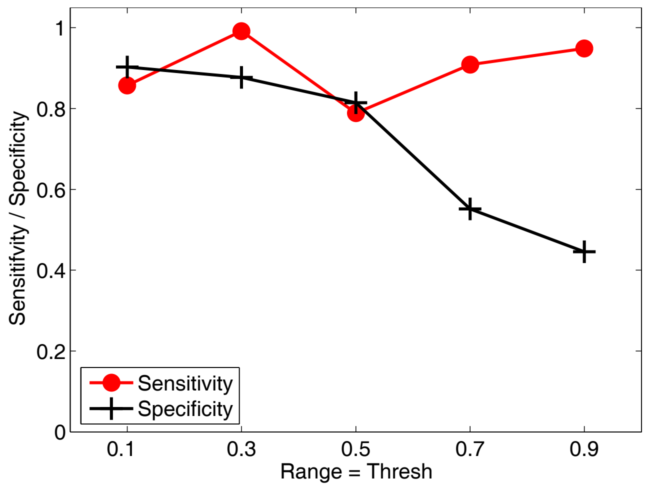

Figure 7: Sensitivity and Specificity for experiment on time series data. Sensitivity refers to the rate of recognizing labels that are associated with time series subsequences. Specificity denotes the rate, with which labels that are attached to random data are identified as insignificant.

4.2 Evaluation on Time Series Data

The algorithm is also tested on time series subsequence data. For this evaluation, subsequences from a particular time series are associated with one item: the label of the time series from which they originate. The significance of the pattern time series label $\Longleftrightarrow$ subsequence vector is tested. This problem can be assumed to be difficult since clusters of time series subsequences are notorious for being independent from the time series that was used for clustering [25].

For this evaluation, 7 data sets from the UCR time series repository [24] are used and one independently collected one: The Ecg series (MIT-BIH Arrhythmia Database: mitdb100) originates from PhysioBank [18]. Descriptions of the data sets from [24] are distributed with the data. Subsequences are extracted using a sliding window approach, and differences between successive data points are considered as the dimensions of the vector space. Windows of length 17 are used resulting in a 16-dimensional vector space. 1000 data points are collected from each time series and a randomly selected subset of 100 is used in the evaluation. The data are combined with the full set of points that can be extracted from the UCR random walk time series (65520 points). For comparison purposes, 7 labels are given to samples of 100 randomly picked points from the random walk time series.

A true positive result indicates a label that is associated with a non-random time series and is identified as significant at the 5% level. If a label is considered insignificant although it belongs to a non-random set, the result is treated as a false negative. A label that is associated with random data and is considered (in)significant at the 5% level is a false positive (true negative). Figure 7 shows the sensitivity (true positives / all positives) and specificity (true negatives /

all negatives) for 50 runs of the data set described in the previous paragraph. It can be seen that the sensitivity is consistently above 0.78. The specificity is greater than 0.8 for range values of 0.5 or less. It appears that for larger range values the result becomes unreliable, possibly due to the overall fluctuations in the data set. Given the challenging nature of time series subsequence data the result can be considered very promising.

4.3 Performance

The algorithm scales linearly with the number of domains. Scaling as a function of domain size is quadratic for large domains, since we construct N_i histograms, each of which represents N_i points. In our data sets most domains are so small that linear contributions related to the χ^2 test play an important role. Figure 8 shows the runtime per domain for the gene expression data set plotted as a function of the number of instances in the domain, N_i. It can be seen that the initial slope is less than 2 indicating that the relationship is not quite quadratic. For larger domain size, the slope increases to a value close to 2. Overall, performance was not a major concern to us since individual runs for the complete set of domains and the entire yeast genome only took minutes, despite the computation-intensive choice of deriving the comparison histogram through averaging rather than through equation (11).

We have not yet tested the algorithm on combinations of domains. The density-histogram-based significance measure does not satisfy downward closure. Hence, a satisfactory performance of an itemset-based evaluation would have to be achieved through setting a support threshold. We are currently working on techniques for vector-item-patterns that are better suited to patterns involving itemsets and multiple vectors.

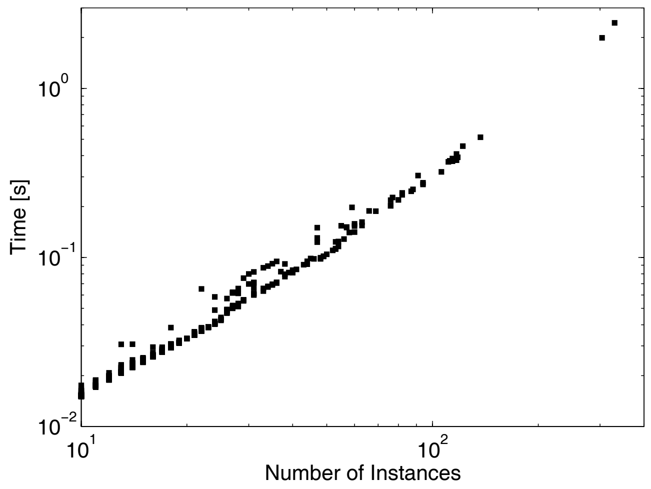

Figure 8: Runtime for evaluation of a single domain for one parameter with expected distribution derived from 50 examples.

5. CONCLUSIONS

We have presented an algorithm for data mining of vector-item patterns based on density histograms. Subsets of the vector data, which are defined by the item data are the basis for the histograms. A density measure is used that considers subspaces. Histograms based on the observed densities at data points are compared with their expected counterparts. A normalization is chosen such that the expected density histograms approximately follow a binomial distribution. Effectiveness and efficiency of our algorithm have been demonstrated using cell-cycle gene expression data and Interpro domain annotations as well as time series subsequence data. We have shown that our algorithm produces results that are consistent among independent experimental data. Comparison with a conventional technique shows that overlap is significant for one of our data sets. For time series subsequence data we were able to show quantitatively that labels associated with non-random data can be recognized with sensitivity and specificity greater than 0.8. This work introduces a new approach toward relating continuous vector data and item data that is potentially valuable in many application areas.

6. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. IDM-0415190. Thanks to Alan Denton for proof-reading the manuscript.

7. REFERENCES

[1] C. Aggarwal. Re-designing distance functions and distance-based applications for high dimensional data. *SIGMOD Record*, 30(1):13–18, 2001.

[2] C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. *Lecture Notes in Computer Science*, 1973, 2001.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data. *Data Mining and Knowledge Discovery Journal*, 11(1), 2005.

[4] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[5] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[6] Z. Bar-Joseph, G. Gerber, T. Jaakkola, D. Gifford, and I. Simon. Continuous representations of time series gene expression data. *J Comput Biol.*, 10(3-4), 2003.

[7] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *RECOMB '02:Proc 6th annual international conference on Computational biology*, New York,NY, 2002.

[8] N. Bolshakova, F. Azuaje, and P. Cunningham. A knowledge-driven approach to cluster validity assessment. *Bioinformatics*, 21(10):2546–7, 2005.

[9] B. M. Bolstad, R. A. Irizarry, M. Astrand, and T. Speed. A comparison of normalization methods for high density oligonucleotide array data based on bias and variance. *Bioinformatics*, 19(2):185–193, 2003.

[10] S. Brin, R. Motwani, and C. Silverstein. Beyond

market baskets: generalizing association rules to correlations. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 265–276, New York, NY, USA, 1997. ACM Press.

[11] Y. Cheng and G. Church. Biclustering of expression data. In *Proc. 8th Int'l Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 93–103, 2000.

[12] R. Chiang, C. E. H. Cencil, and E.-P. Lim. Linear correlation discovery in databases: a data mining approach. *Data and Knowledge Engineering*, 53:311–337, 2005.

[13] D. Chudova, C. Hart, E. Mjolsness, and P. Smyth. Gene expression clustering with functional mixture models. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2003.

[14] S. G. Database. Interproscan results using s. cerevisiae protein sequences ftp://genome-ftp.stanford.edu/pub/yeast/sequence_similarity/domains/domains.tab.

[15] A. Denton and A. Kar. Finding differentially expressed genes through noise elimination. In *Proc. of the Data Mining for Biomedical Informatics workshop in conjunction with the 7th SIAM Int'l Conf. on Data Mining*, Minneapolis, MN, April 2007.

[16] J. Ernst, G. Nau, and Z. Bar-Joesph. Clustering short time series gene expression data. *Bioinformatics*, 21(Supplement 1), 2005.

[17] B. Gao, O. Griffith, M. Ester, and S. Jones. Discovering significant opsm subspace clusters in massive gene expression data. In *2006 ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, Philadelphia,PA, 2006.

[18] A. L. Goldberger et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000 (June 13). Circulation Electronic Pages: [http://circ.ahajournals.org/cgi/content/full/101/23/e215].

[19] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.

[20] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Trans. Knowl. Data Eng.*, 16(11):1370–1386, 2004.

[21] P. Jonsson, K. Laurio, Z. Lubovac, B. Olsson, and M. L. Andersson. Using functional annotation to improve clusterings of gene expression patterns. In *Proc. of the 6th Joint Conference on Information Science*, pages 1257–1262, 2002.

[22] K. Kailing, H. Kriegel, P. Kroeger, and S. Wanka. Ranking interesting subspaces for clustering high dimensional data. In *Proceedings of the PKDD Conference*, pages 241–252, 2003.

[23] S. Kaski, J. Sinkkonen, and J. Nikkilä. Clustering gene expression data by mutual information with gene function. In *Proc. of the Int'l Conference on Artificial Neural Networks (ICANN)*, pages 81–86, 2001.

[24] E. Keogh and T. Folias. The ucr time series data mining archive, accessed 2003. [http://www.cs.ucr.edu/∼eamonn/TSDMA/index.html].

[25] E. Keogh, J. Lin, and W. Truppel. Clustering of time series subsequences is meaningless: implications for previous and future research. In *Proceedings of the IEEE Int. Conf. on Data Mining*, pages 115–122, Melbourne, FL, 2003.

[26] MATLAB. Documentation http://www.mathworks.com/access/helpdesk/help/toolbox/stats/chi2gof.html, accessed 02/07.

[27] N. Mulder1, R. Apweiler, and T. Attwood. New developments in the interpro database. *Nucleic Acids Research*, 35:D224–228, 2007.

[28] R. Rastogi and K. Shim. Mining optimized support rules for numeric attributes. *Information Systems*, 26(6):425–444, 2001.

[29] V. Roth, M. Braun, T. Lange, and J. Buhmann. A resampling approach to cluster validation. In *Computational Statistics (COMPSTAT'02)*, 2002.

[30] P. Spellman. Yeast cell cycle analysis project, http://cellcycle-www.stanford.edu/, accessed 2007.

[31] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.

[32] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. 1996 ACM SIGMOD Int'l Conf. on Management of Data*, pages 1–12, Montreal, Quebec, Canada, 4–6 1996.

[33] F. Yates. Contingency table involving small numbers and the χ^2 test. *Journal of the Royal Statistical Society (Supplement)*, 1:217–235, 1934.

[34] K. Yeung, D. R. Haynor, and W. L. Ruzzo. Validating clustering for gene expression data. *Bioinformatics*, 17, 2001.

[35] G. Yona, W. Dirks, S. Rahman, and D. Lin. Effective similarity measures for expression profiles. *Bioinformatics*, 22(13):1616–1622, 2006.

An Ensemble based Bayesian Network Learning Algorithm on Limited Data*

Feng Liu
Beijing University of Posts and Telecommunications
Beijing, China
lliufeng@hotmail.com

Fengzhan Tian
Beijing Jiaotong University
Beijing, China
fztian@mail.bjtu.edu.cn

Qiliang Zhu
Beijing University of Posts and Telecommunications
Beijing, China
zhuqiliang@tom.com

ABSTRACT
In recent years, Bayesian network (BN) has become successful tool for analyzing model structures in real-life domains. We present an efficient approach to produce a more accurate Bayesian network from limited datasets. Our approach incorporates ensemble learning into BN learning algorithms. Based on the Markov condition of BN learning, our ensemble based BN learning approach proposes a novel sampling technique and components integration technique. The experimental results reveal that our ensemble based BN learning approach can achieve an improved result compared with individual BN learning approach in terms of accuracy on limited datasets.

Categories and Subject Descriptors
H.4 [**Knowledge discovery and data mining**]: Miscellaneous

General Terms
Knowledge

Keywords
Bayesian network, Ensemble method, Sampling, Integrating

1. INTRODUCTION
Bagging [5] is one of the most popular methods of ensemble learning. Bagging method uses Efron's bootstrap sampling [4]. On each run, Bagging method presents the component learner with a training dataset that consists of a sample of m cases drawn randomly with replacement from the original training dataset of m cases.[5] Bootstrap sampling is a powerful tool for estimating various properties of a given statistic in statistics. It quickly gained popularity in model selection. When learning the structure of a graphical model from limited datasets, such as the gene-expression datasets, Bagging method has been applied to explore model structures.[6],[7],[8],[12],[3]

However, Bagging method has some disadvantages over BN learning. This is because bootstrap sampling method involves a few problems (an overview in [4]). For example, as described by Steck and Jaakkola in [12], the bootstrap method produces the *spurious dependences* problem which can cause the graphical models learned from the datasets be biased towards complex models, especially from limited datasets. As a result, many extra edges may be present in the learned model structures, and the confidence in the presence of the extra edges may be overestimated. Furthermore, even if the results of Bagging method are asymptotically converge to the true BN by sampling several hundreds of times in some cases, some convergent conditions (discussed in [6]) for the true distribution are needed. These conditions may be unsatisfied in some other cases.

In this paper, we propose a novel sampling method called Root Nodes based Sampling (RNS) method and components integration method. Since the distributions over the sampling datasets obtained using RNS method and the true BN satisfy the Markov condition [11], RNS method reduces the structural inconsistencies between the Bayesian networks learned from the sampling datasets and the true BN. Furthermore, RNS method does not need any convergent condition in the true distribution. In addition, our components integration method futher improves the accuracy in Bayesian network induction on limited datasets.

The rest of this paper is organized as follows. In Section 2, we propose the process of our ensemble based BN learning approach. We present the details of our sampling technique and components integration technique in sections 3 and 4. In section 5, experimental results are compared and analyzed. Finally, we conclude our work in section 6.

2. ENSEMBLE BN LEARNING OVERVIEW
Ensemble methods are the learning algorithms that a component learner is trained multiple times for the same task, and these trained outcomes (components) are combined for

dealing with future instances.[5]

Given the original training dataset D, our ensemble based BN learning approach applies sampling technique to generate several sampling datasets D_i. Then, the component learner learns a component (BN) BN$_i$ from each sampling dataset D_i. Finally, using components integration technique, these learned components are combined into a more accurate BN.

3. ROOT NODES BASED SAMPLING

Constructing a good ensemble for BN learning involves holding the accuracy and increasing the diversity among the component learners. The idea behind RNS method is based on the following 2 facts:

1. The distributions over the sampling datasets obtained using RNS method and the true BN satisfy the Markov condition which determines a set of independence relations. So, RNS method can avoid the generation of extra edges in the learned components (Bayesian networks), and thus improves the accuracy for components (BNs) learning (compare with Bagging method).
2. The joint marginal probability distributions of the combination of some nodes vary with the different sampled sub-datasets. So, RNS method can increase the diversity for components learning. Combined with integration method, this can alleviate the unreliability of the CI measures and the score functions during learning components from limited datasets.

The detail of RNS method is shown in Fig.1.

```
1) Search root nodes $H_i$ ($i$=1,...,$L$) from the original training dataset $D$ ;
2) Compute the marginal probability tables $P(H_i)$, $1 \le i \le L$ of each found root node;
3) Construct the marginal probability tables $P(H_i, H_j)$, $1 \le i,j \le L$ for pairs of the found root nodes, where $P(H_i, H_j)=P(H_i)P(H_j)$;
4) For every marginal probability table $P(H_i, H_j)$, $1 \le i,j \le L$
     For every possible value $(h_i, h_j)$ of the root nodes pair $\{H_i, H_j\}$, $(h_i, h_j) \in (H_i, H_j)$
       ● Select the cases which satisfy the condition $\{H_i \ne h_i \wedge H_j \ne h_j\}$ from the original training dataset $D$ to form the sub-dataset $Ds_{\{H_i \ne h_i \wedge H_j \ne h_j\}}$ ;
       ● Combine the original training dataset $D$ and the sub-dataset $Ds_{\{H_i \ne h_i \wedge H_j \ne h_j\}}$ into the sampling dataset $D_{\{H_i \ne h_i \wedge H_j \ne h_j\}}$ ;
     EndFor
     Obtain the group $Gr(H_i, H_j)$ consisting of the sampling datasets $D_{\{H_i \ne h_i \wedge H_j \ne h_j\}}$, where $(h_i, h_j) \in (H_i, H_j)$;
   EndFor
5) Return the groups $Gr(H_i, H_j)$, $1 \le i,j \le L$ of the sampling datasets.
```

Figure 1: Root nodes based sampling method

Firstly, assume that the algorithm found some root nodes in step 1 and computed the marginal probability tables of the found root nodes. Then, the marginal probability tables of all the pairs of the found root nodes are constructed in step 3. In step 4, RNS method firstly sub-samples the original training dataset D by limitting the values of the pair of the found root nodes $\{H_i, H_j\}$ to $\{H_i \neq h_i \wedge H_j \neq h_j\}$ and obtains the sub-dataset $Ds_{\{H_i \neq h_i \wedge H_j \neq h_j\}}$. Then, the original training dataset D and the sub-dataset $Ds_{\{H_i \neq h_i \wedge H_j \neq h_j\}}$ are combined into the sampling dataset $D_{\{H_i \neq h_i \wedge H_j \neq h_j\}}$. RNS method repeats the 2 statements in step 4 for each vaule $\{h_i, h_j\}$ of each pair $\{H_i, H_j\}$ of the found root nodes and obtains all the sampling datasets. RNS method groups the sampling datasets into the groups $Gr(H_i, H_j)$,$1 \le i, j \le L$ according to the pair $\{H_i, H_j\}$,$1 \le i, j \le L$ of the found roots. Finally, RNS method returns these groups of sampling dataset.

3.1 Correctness

Assume that the original training dataset D is data faithful to the true BN G.[11] Assume that the root nodes found by RNS method are the root nodes in the true BN G.

PROPOSITION 1. *If we learn a BN from the sampling dataset obtained from the original training dataset D using Root Nodes based Sampling method, then we can get all the Markov independences implied by the true Bayesian network G.*

PROOF. Assume that the nodes A and B are root nodes in G. The range of values for A is $\{a_1, \ldots, a_{m_a}\}$. The range of values for B is $\{b_1, \ldots, b_{m_b}\}$. The sampling dataset $D_{\{A \neq a_1 \wedge B \neq b_1\}}$ is obtained from D using RNS method. The node E is a non-parent and non-descendant node of C in the true Bayesian network G. Π_C is the parents set of C in the true BN G. There exists the Markov independence $Ind(C, E \mid \Pi_C)$ in the true BN G.

Assume that P denotes the distribution faithful to the true BN and P' denotes the distribution over the sampling dataset $D_{\{A \neq a_1 \wedge B \neq b_1\}}$.

We take 3 cases to consider whether there exists $Ind(C, E \mid \Pi_C)$ in P' over the sampling dataset $D_{\{A \neq a_1 \wedge B \neq b_1\}}$.

1. In the case that Π_C contains the root nodes A and B. Assume that $\Pi_C = \{A, B, H\}$. We know that $Ind(C, E \mid \{A, B, H\})$ in the true BN. H is a parent node of C.

   From the sampling dataset $D_{\{A \neq a_1 \wedge B \neq b_1\}}$, we can infer the formula:

   When $(A \neq a_1)$ and $(B \neq b_1)$,

   $$P'(c \mid E = e, H = h, A = a, B = b) = N'_{cehab}/N'_{ehab}$$
   $$= 2N_{cehab}/2N_{ehab} = P'(c \mid h, a, b)$$

   When $(A = a_1)$ and $(B = b_1)$,

   $$P'(c \mid E = e, H = h, A = a_1, B = b_1)$$
   $$= N'_{ceha_1b_1}/N'_{eha_1b_1}$$
   $$= N_{ceha_1b_1}/N_{eha_1b_1} = P'(c \mid h, a_1, b_1)$$

   According to the definition of conditional independence [11], we can infer that there exists $Ind(C, E \mid A, B, H)$ in the distribution P'.

2. In the case that Π_C contains one of the two root nodes A and B. Assume that $\Pi_C = \{A, H\}$. We know that

$Ind(C, E \mid \{A, H\})$ in the true BN. H is a parent node of C.

From the sampling dataset $D_{\{A \neq a_1 \wedge B \neq b_1\}}$, we can infer the formula:

When $(A \neq a_1)$,

$$P'(c \mid E = e, H = h, A = a) = N'_{ceha}/N'_{eha}$$
$$= \sum_b N'_{cehab} / \sum_b N'_{ehab} = \sum_b N_{cehab} / \sum_b N_{ehab}$$

When $(A = a_1)$,

$$P'(c \mid E = e, H = h, A = a) = N'_{ceha}/N'_{eha}$$
$$= \frac{N_{ceha_1b_1} + \sum_{b \neq b_1} 2N_{ceha_1b}}{N_{eha_1b_1} + \sum_{b \neq b_1} 2N_{eha_1b}}$$

According to the following formula: $P(c \mid E = e, H = h, A = a, B = b) = N_{cehab}/N_{ehab} = P(c \mid h, a)$, we can infer that $P'(c \mid E = e, H = h, A = a) = P(c \mid H = h, A = a)$.

Using the above method, we can also infer that $P'(c \mid H = h, A = a) = P(c \mid H = h, A = a)$.

So, we infer that $P'(c \mid E = e, H = h, A = a) = P'(c \mid H = h, A = a)$.

According to the definition of conditional independence, we can infer that there exists $Ind(C, E \mid A, H)$ in the distribution P'.

3. In the case that Π_C contains none of the two root nodes A and B. Assume that $\Pi_C = \{H\}$. We know that $Ind(C, E \mid H)$ in the true BN. H is a parent node of C.

   From the sampling dataset $D_{\{A \neq a_1 \wedge B \neq b_1\}}$, we can infer the formula:

$$P'(c \mid E = e, H = h) = N'_{ceh}/N'_{eh}$$
$$= \frac{\sum_{a \neq a_1 \wedge b \neq b_1} 2N_{cehab} + N_{ceha_1b_1}}{\sum_{a \neq a_1 \wedge b \neq b_1} 2N_{ehab} + N_{eha_1b_1}}$$

   According to the following formula: $P(c \mid E = e, H = h, A = a, B = b) = N_{cehab}/N_{ehab} = P(c \mid h)$, we can infer that $P'(c \mid E = e, H = h) = P(c \mid H = h)$.

   Using the above method, we can also infer that $P'(c \mid H = h) = P(c \mid H = h)$.

   So, we infer that $P'(c \mid E = e, H = h) = P'(c \mid H = h)$.

   According to the definition of conditional independence, we can infer that there exists $Ind(C, E \mid H)$ in the distribution P'.

According to the cases 1, 2 and 3, we infer that Proposition 1 is correct. □

3.2 Root Nodes Search Method

The idea behind our root nodes search method is based on the following rules.

Rule 1. *If $Ind(X, Z \mid NULL)$ and $Dep(X, Z \mid Y)$, then we can add a head-to-head pattern among X,Y and Z, that is $X \longmapsto Y$ and $Z \longmapsto Y$.*

Rule 2. *Given $Ind(X, Z \mid NULL)$, $Dep(X, Y \mid NULL)$, $Dep(Y, Z \mid NULL)$, $X \longmapsto Y - Z$, if $Ind(X, Z \mid Y)$, then we can direct the edge $Y - Z$ to be $Y \rightarrow Z$, that is $X \longmapsto Y \rightarrow Z$.*

The search method is shown in Fig.2. Note: A maximal clique is the clique which is not contained by any other clique. δ denotes the upper bound on the number of nodes in a group of cliques.

1) Compute order-0 CI tests for each pair of the nodes and build the undirected graph UG_0, where if *Dep*(*X*,*Y* |*NULL*), then there exists an edge *X*—*Y* in UG_0;
2) Compute order-1 CI tests for any three nodes *X*, *Y* and *Z* that in UG_0, *X*—*Y*, *Y*—*Z*, and *X* and *Z* are not directly connected, and apply Rules 1 and 2 to direct the edges *X*—*Y* and *Y*—*Z* in UG_0;
3) Find the maximal cliques $\{C_1,...,C_k\}$ consisting of the nodes which have no inarcs in UG_0;
4) Divide the maximal cliques into groups $\{GrC_1,...,GrC_t\}$ consisting of the maximal cliques each of which has common nodes with at least one other clique in the same group;
5) Order the groups of cliques $\{GrC_1,...,GrC_t\}$ by the number of nodes in ascendant order and prune the groups GrC_j of cliques where $\| GrC_j \| > \delta$;
6) For each group GrC_i of cliques, use the exhaustive search & score method over the original training dataset to learn the root node in the group of cliques;
7) Detect and delete pseudo root nodes.

Figure 2: Root nodes search method

Assume that the original training dataset is data faithful to the true BN.[11] We consider the following two possibilities for the node X which is not root node in the true BN.

1. There exist two or more root nodes that have directed paths to node X in the true BN.

   Assume there are two root nodes $R1$ and $R2$ that have directed paths to X in the true BN. After step 2 in Fig.2, there are 3 possible cases shown in Fig.3 for X in UG_0.

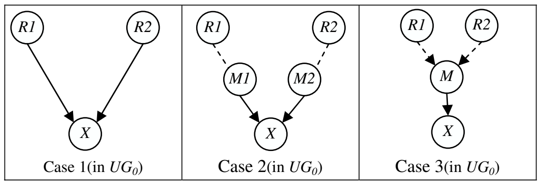

Figure 3: 3 cases for X in UG_0 after step 2

   Therefore, we see that there must be at least one directed edge to X in UG_0 after step 2 in Fig.2. When there are three or more root nodes $R1$ and $R2$ that have directed paths to X in the true BN, we can also infer that there must be at least one directed edge to X in UG_0 after step 2 in Fig.2.

2. There exists only one root node $R0$ that has directed paths $R0 \longmapsto X$ to node X in the true BN.

   After step 3 in Fig.2, we find a few maximal cliques each of which contains at most one root node and the non-root nodes that has directed paths to them from one and only one root node in the true BN. X must be contained in some maximal cliques after step 3.

In step 4 in Fig.2, the cliques that has common nodes with at least one other clique in the same group are divided into a group. We know that each non-root node in a clique has and only has one root node that has directed paths to it in the true BN. So, each group of cliques only contains one root node and each root node only exists in one group of cliques. Therefore, if the original training dataset is data faithful to the true BN, then we can find all the root nodes in the true BN after step 6 in Fig.2.

However, the faithfulness assumption often can not be satisfied on limited datasets. Some results after step 6 in Fig.2 may be not root nodes in the true BN, which are called "pseudo root node". These exceptions often occur when there are many nodes in a group of cliques.

We take 2 steps to solve the pseudo root nodes problem. The first step is to limit the number of nodes in a group of cliques (see step 5 in Fig.2). We discard the groups of cliques in which the number of nodes are larger than a given threshold δ. The step makes exhaustive search practically feasible and greatly enhance the accuracy of our search method. The other step is to detect pseudo root nodes and delete them (step 7 in Fig.2). The detection for pseudo root nodes is based on the fact that the ancestors of a node in a group of cliques must be in the same group of cliques. So, we applies the following rule to detect the pseudo root nodes.

Assume that GrC_i is a group of cliques. $X, Y, R \in GrC_i$. R is the root node found after step 6 in Fig.2.

RULE 3. *If $\exists S$, $S \subset GrC_i$, $R \notin S$, $Ind(X, Y \mid S)$ and $Dep(X, Y \mid S, R)$, then the found root node R is pseudo root node.*

Even if our method finally learned a tiny number of pseudo root nodes from small training datasets, we can prove that if RNS method is used by the pseudo root node, it does not change the Markov independences of other nodes implied by the true BN, except for its ancestor nodes. This is one cause that we limit the number of nodes in a group of cliques. Moreover, Our intergroup integration method discussed in Section 4.2 further eliminated the effect of the pseudo root nodes problem. Finally, our search method does not have to find all the root nodes in the true BN, it is enough to find several root nodes in the true BN for our ensemble based BN learning in terms of accuracy.

4. BNS INTEGRATION METHOD

Components Integration is the last step of our ensemble based BN learning algorithm. Our integration method includes 2 parts: integrating the components (BNs) learned from the sampling datasets in the same group; integrating the intergroup undirected networks generated in the first part.

Assume that we obtained m groups $Gr_i(i = 1, \ldots, m)$ of sampling datasets using RNS method on the original training dataset D and each group $Gr_i(i = 1, \ldots, m)$ contains L_i sampling datasets.

Assume that we learned m groups $GBN_i(i = 1, \ldots, m)$ of components (BNs) using selected BN learner. Each group GBN_i consists of the L_i BNs $\{BN_{i1}, \ldots, BN_{iL_i}\}$ learned from the L_i sampling datasets in the group Gr_i using selected BN learner.

4.1 Intragroup BNs Integration

According to Proposition 1, we see that each Bayesian network learned from the sampling datasets obtained using RNS method contains all the Markov independences implied by the true BN, which avoids the advent of extra edges in the learned BNs. Moreover, almost each edge in the true BN can be included in most of the BNs of a group $GBN_i(i = 1, \ldots, m)$. So, after the intragroup BNs integration, we can get an undirected network which nearly includes all the edges in the true BN. The process is shown in Fig.4.

```
/* Create the undirected networks $UG_i$ (i=1,...,m) */
For each group $GBN_i$ (i=1,...,m)
  1. For each edge $\tilde{e}$ which directed arc is in the BNs
     $BN_{i1}, \ldots, BN_{iL_i}$ , compute occurrence rate $bs(\tilde{e})$ ;
  2. Prune the edges whose probabilities are less than 1/2 ;
  3. Obtain a undirected network $UG_i$ in which every edge $\tilde{e}$
     has a probability table $g_i(\tilde{e})$;
```

Figure 4: Intragroup BNs integration method

For any edge $\widetilde{e}$ which directed arc e is in the BNs $BN_{i1}, \ldots,$ BN_{iL_i}, consider the quantity

$$bs(\widetilde{e}) = \frac{1}{L_i} \sum_{j=1}^{L_i} 1\left(e^{\leftarrow} \in BN_{ij} \| e^{\rightarrow} \in BN_{ij}\right).$$

In the undirected network UG_i, every edge $\widetilde{e}$ has a probability table $g_i(\widetilde{e})$ which represents the probabilities $g_i(e)$ of directed arcs $e \in \{e^{\leftarrow}, e^{\rightarrow}\}$: $g_i(e) = \frac{1}{L_i} \sum\limits_{j=1}^{L_i} 1\left(e \in BN_{ij}\right), e \in \{e^{\leftarrow}, e^{\rightarrow}\}$.

Note: the values of function $1(\ldots)$ are 1(TRUE)=1 and 1(FALSE)=0.

4.2 Intergroup UNs Integration

After the intragroup BNs integration, we obtained m undirected networks (UNs) $UG_i(i = 1, \ldots, m)$ which approximate the true BN in the form of undirected network to maximal extent.

As discussed in the above subsection, most of the edges in the true BN can be included in most of the undirected networks. So, our intergroup integration approach firstly uses the majority voting to obtain the most probable edges which exist in the true BN and generate the final undirected network UG. Then, our approach directs the edges in UG and obtain an initial result Bayesian network. Finally, we insert the rest edges in the learned UNs according to the maximal score principle and the acyclic property of BN and get the result BN. The process is shown in Fig.5.

```
/* Create the undirected network UG */
1) For each undirected edge $\tilde{e}$ in $UG_i$ ($i$=1,...,$m$), compute
   occurrence rate $us(\tilde{e})$;
2) Create a undirected network UG consisting of the undirected
   edges, which occurrence rate $us(\tilde{e})$ are no less than 1/2, in
   the $UG_i$ ($i$=1,...,$m$);
/* Direct the UG to get the initial BN*/
3) Compute the probability $pw(e)$ of each possible directed arc
   $e$ which undirected edge is in the UG;
4) Direct the edges in the UG using causality inference rules or
   the probability $pw(e)$ to obtain an initial BN G;
/* Insert the rest arcs */
5) For the possible directed arcs {e} which undirected edges
   {$\tilde{e}$} are not in UG, but in $UG_i$ ($i$=1,...,$m$), apply greedy
   search & Bayesian score function and the acyclic property
   of BN to insert into the Bayesian network G;
6) Return the result Bayesian network G
```

Figure 5: Intergroup UNs integration method

The occurrence rate $us(\widetilde{e})$ of each undirected edge $\widetilde{e}$ in the $UG_i(i = 1, \ldots, m)$ is the quantity, $us(\widetilde{e}) = \frac{1}{m}\sum_{i=1}^{m} 1\left(\widetilde{e} \in UG_i\right)$.

Directing edges is conducted according to causality rules[11] by identifying V-structures, i.e., non-adjacent parents having a common child, directing the relevant edges, and applying additional rules to further direct edges until no more edges can be directed. For the rest undirected edges in UG, we consider the quantity $pw(e) = \frac{1}{m}\sum_{i=1}^{m} g_i(e)1\left(\widetilde{e} \in UG_i\right)$, $e \in \{e^{\leftarrow}, e^{\rightarrow}\}$. If $pw(e^{\leftarrow}) > pw(e^{\rightarrow})$, then it is more probable that $e^{\leftarrow}$ exists in the true BN than $e^{\rightarrow}$ does, and then we direct $\widetilde{e}$ to be $e^{\leftarrow}$. Vice versa.

On limited datasets, it is possible that a small number of edges in the true BN are only included in a small number of UNs learned in the first part of our integration method. So, we take greedy search & Bayesian score method to insert the rest edges in the UNs into the initial Bayesian network and obtain the result Bayesian network.

5. EXPERIMENTAL RESULTS

BN learner is an individual BN learning algorithm and a building block of ensemble based BN learning. Normally, it needs to be efficient computationally. Therefore, we selected OR algorithm [10] and TPDA algorithm [2] as our BN learners.

We implemented OR algorithm [10], OR-BSV algorithm, OR-RNS algorithm, TPDA algorithm [2], TPDA-BSV algorithm, and TPDA-RNS algorithm.

OR-BSV and TPDA-BSV algorithms use Bagging method and the simple voting integration method, where m bootstrap sampling datasets $D_1, \ldots, D_m$ are generated from the original training dataset D and a component (BN) BN_i is obtained from each D_i, an integrated result BN BN_r is built from $BN_1, \ldots, BN_m$. The structure of BN_r is determined through majority voting, where an edge exists if and only if such an edge exists in majority BNs.[1]

Tests were run on a PC with Pentium4 1.5GHz and 1GB RAM. The operating system was Windows 2000. 3 Bayesian networks (Alarm, Barley, Hailf(inder)) were used. Table 1 shows the characteristics of these networks. The characteristics include the number of nodes, the number of arcs, the number of root nodes, the upper bound on the parents/children number of a node (Max In/Out-degree), and the minimal/maximal number of node values(Domain Range).

Table 1: Bayesian Networks

BN	Nodes Num	Arcs Num	Roots Num	Max In/ Out-Degree	Domain Range
Alarm	37	46	12	4/5	2-4
Barley	48	84	10	4/5	2-67
Hailf	56	66	17	4/16	2-11

We performed experiments on the training datasets with 200, 500, 1000 instances. For each network and sample size, we sampled 10 original training datasets. We applied bootstrap sampling with 200 times in OR-BSV and TPDA-BSV algorithms. Let $\delta = 5$ in Fig.2.

For each network and sample size, we recorded the average results of the number of the true root nodes and the number of the pseudo root nodes found by our root nodes search method from the original training datasets. From the results in Tables 2-4, we can see that our root nodes search method is very efficient on all the original training datasets. 'Posi-RNs' denotes the number of true root nodes learned by our search method. 'Pseudo-RNs' denotes the number of pseudo root nodes learned by our search method.

Table 2: Alarm Net

SIZE	Posi-RNs	Pseudo-RNs
200	7	1
500	9	1
1000	10	0

Table 3: Barley Net

SIZE	Posi-RNs	Pseudo-RNs
200	6	2
500	7	1
1000	8	1

Table 4: Hailf Net

SIZE	Posi-RNs	Pseudo-RNs
200	11	0
500	14	0
1000	15	0

We compared the accuracy of Bayesian networks learned by these algorithms according to the average BDeu score. The BDeu score corresponds to the posteriori probability of the structure learned.[9] The BDeu scores in our experiments were calculated on a seperate testing dataset sampled from the true BN containing 20000 instances. For each network

Table 5: Average BDeu(Alarm-OR)

SIZE	OR	OR-BSV	OR-RNS
200	-17.38±0.91	-16.77±2.13	-15.96±1.49
500	-15.10±0.22	-14.93±0.95	-14.17±0.55
1000	-14.82±0.16	-14.61±0.39	-13.75±0.23

Table 6: Average BDeu(Barley-OR)

SIZE	OR	OR-BSV	OR-RNS
200	-87.51±3.45	-86.92±6.86	-85.61±5.13
500	-82.35±2.14	-81.74±4.26	-79.93±3.08
1000	-78.97±1.93	-78.26±3.38	-77.03±2.42

Table 7: Average BDeu(Hailf-OR)

SIZE	OR	OR-BSV	OR-RNS
200	-76.14±2.69	-75.83±5.64	-74.77±3.79
500	-73.98±1.38	-73.23±2.82	-72.35±2.31
1000	-73.21±0.96	-73.07±1.65	-71.91±1.35

Table 8: Average BDeu(Alarm-TPDA)

SIZE	TPDA	TPDA-BSV	TPDA-RNS
200	-22.47±2.78	-21.91±5.96	-20.81±3.79
500	-18.10±1.53	-17.52±3.03	-16.90±2.01
1000	-15.43±0.98	-15.00±1.73	-14.52±1.38

Table 9: Average BDeu(Barley-TPDA)

SIZE	TPDA	TPDA-BSV	TPDA-RNS
200	-100.92±6.29	-107.57±12.37	-104.87±7.91
500	-103.79±5.38	-107.92±11.92	-107.89±6.83
1000	-111.07±4.69	-106.55±10.95	-106.96±6.14

Table 10: Average BDeu(Hailf-TPDA)

SIZE	TPDA	TPDA-BSV	TPDA-RNS
200	-103.13±4.91	-102.17±8.16	-99.72±6.16
500	-101.38±4.27	-99.69±6.74	-98.39±5.38
1000	-97.38±3.30	-96.51±5.57	-95.61±4.27

and sample size, we recorded the average result and standard deviation of the BDeu scores of the BNs learned from the original training datasets by these algorithms. Tables 5-10 report the results.

There are several noticeable trends in the average results. Firstly, as expected, as the number of instances grow, the quality of learned Bayesian network improves, except to TPDA, TPDA-BSV and TPDA-RNS for Barley network. It is due to that constraint-based method is unstable for small datasets (200,500). At the same time, we can see that TPDA-BSV and TPDA-RNS algorithm improve the stability of TPDA. Secondly, RNS based BN learning algorithms OR-RNS and TPDA-RNS are almost better than or at least equal to the individual BN learning algorithms in terms of accuracy on limited datasets. Thirdly, in most cases, RNS based BN learning algorithms have better performance than Bagging based BN learning algorithms.

From the results of standard deviation, we can see the following trends. Firstly, as the number of instances grow, the variability among the BNs learned from the original training datasets reduces for all the 4 networks and all the 6 algorithms. Secondly, Bagging based BN learning algorithms produced much larger standard deviation on small datasets(200, 500) because the *spurious dependences* problem of bootstrap methods discussed in section 1. However, RNS based BN learning algorithms produced less standard deviation, especially on small datasets(200, 500). So, this confirms that RNS method alleviates the *spurious dependences* problem of bootstrap methods on small datasets to some extent. In the context of model selection, this also makes us limit the range of approximately true BNs more definitely on small datasets.

6. CONCLUSION

In this paper, we proposed a novel sampling technique and components integration technique to incorporate ensemble based learning into BN learning on limited datasets. Our results are encouraging in that they indicate that our ensemble based BN learning algorithms achieved more accurate BNs than individual BN learning algorithms on limited datasets.

7. REFERENCES

[1] L. M. Campos, J. Gamez, and J. M. Puerta. Graphical models to causal discovery from data. In *PGM2002 Proceedings*, pages 45–53, 2002.

[2] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning belief networks form data: An information theory based approach. *Artificial Intelligence*, 137(1-2):43–90, 2002.

[3] H. H. Dai, G. Li, and Z. H. Zhou. Ensembling mml causal discovery. In *PAKDD2004 Proceedings*, pages 260–271, 2004.

[4] A. C. Davison and D. V. Hinkley. *Bootstrap methods and their application*. Cambridge Press, NY, 1997.

[5] T. G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):745–770, 1997.

[6] N. Friedman, M. Goldszmidt, and A. Wyner. Data analysis with bayesian networks: A bootstrap approach. In *UAI1999 Proceedings*, pages 196–205, 1999.

[7] N. Friedman, M. Goldszmidt, and A. Wyner. On the application of the bootstrap for computing confidence measures on features of induced bayesian networks. In *AIStatistics1999 Proceedings*, pages 197–202, 1999.

[8] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620, 2000.

[9] D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[10] A. W. Moore and W.-K. Wong. Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In *ICML03 Conference Proceedings*, pages 552–559, 2003.

[11] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. MIT Press, USA, 2000.

[12] H. Steck and T. S. Jaakkola. Bias-corrected bootstrap and model uncertainty. In *NIPS03 Proceedings*, 2003.

Combining Mining Results from Multiple Sources in Clinical Trials and Microarray Applications

Fatih Altiparmak[1], Ozgur Ozturk[1],Selnur Erdal[2], Hakan Ferhatosmanoglu[1],Donald C. Trost[3]
[1]Department of Computer Science and Engineering, The Ohio State University
[2]The Ohio State University Medical Center
[3]Pfizer Inc., Global Research and Development
Email: altiparm@cse.ohio-state.edu

ABSTRACT

The current time series data mining methods generally assume that the series have equal dimensionality, equal length time intervals, and extend over equal length time periods. However,these assumptions are not valid for many real data sets involving multiple data sources. We propose a mining framework to gather high quality information from heterogeneous and multi-dimensional time series data and apply it to two important classes of biomedical data mining applications: Clinical Trials and Microarray Gene Expressions. The framework has two steps: (1) Significant and homogeneous subsets of data (e.g., data generated by similar sources) are selected and analyzed using the mining algorithm of interest, (2) The information gathered in the first step is combined by identifying common (or distinctive) patterns. We demonstrate this framework in the Clinical Trials application by clustering the time series of blood ingredients, *analytes*, of each patient as the first step. The common patterns across the clusters are then identified as highly correlated analyte groups. The patterns can also be utilized to identify a global panel of analytes, which is shown to contain are presentative from each biological group. In the Microarray application, the time series of gene expression from heterogeneous sources of microarray data and mining results of different distance metrics are grouped,and the common patterns over these clusters are mined to extract strong rules for gene expression. The quality of the results is shown to improve with the number of data sets and/or metrics used for mining.

Introduction

With the increasing availability of low cost, high throughput methods introduced by the advancing technology in biosciences and the sharing of the data resulting from such experiments on ubiquitous high-bandwidth networks, we're facing an abundance of data. This is an immense opportunity for speeding up advancement of biosciences; however it also presents us with equally immense challenges such as high-dimensionality, heterogeneity, non-uniformity, insufficient length, unequal interval sizes, variable sampling rates, different lengths, differing calibrations and sensitivity, and noise levels of data sources as depicted on Figure 1 on the next page.

The current time series data mining methods generally assume that the series are sufficiently long, have equal length time intervals, and/or extend over equal length time periods. However, these assumptions are not valid for many real data sets, like biomedical databases. Hence, these data sets can not be mined globally. In case the data is grouped by the source, the data of each source is homogenous (Figure 2). The current approaches can then be applied on each data source separately. But, there will be as many result sets as number of data sources. So, the mining results gathered from each source should then be combined.

We develop a two step mining framework, Information mining (depicted in Figure 2), to gather high quality information from heterogeneous and high-dimensional time series data. In the first step, significant and homogeneous subsets of data (e.g., data generated by similar sources) are selected and analyzed using the mining algorithm of interest. In the second step, the information gathered in the first step is combined by identifying common (or distinctive) patterns over the results of mining of the subsets. Two datasets are identified for the example run shown in Figure 2. In Step 1, the attributes of each subset are clustered separately. The resulting clusterings are {{u, x}, {w, y}} for dataset 1 and {{w, y}, {v, x}} for the second dataset. In Step 2 frequently occurring clusters in Step 1 are extracted. In this case, the framework returns {w, y} set as it appears in both clusterings.

In the Information Mining (IM) Framework, alternative DM techniques can also be utilized in Step 1 such as declustering [1] and outlier detection algorithms [2]. The information sought in Step 2 can also differ; such as finding rarely occurring clusters rather than frequently occurring clusters. In the next section, we will examine two case studies that we applied information mining; clinical trials and microarray datasets.

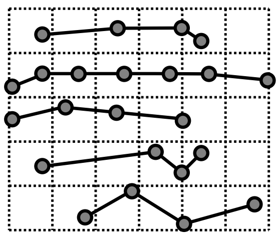

Figure 1: Heterogeneous data that is not compatible with conventional data mining techniques

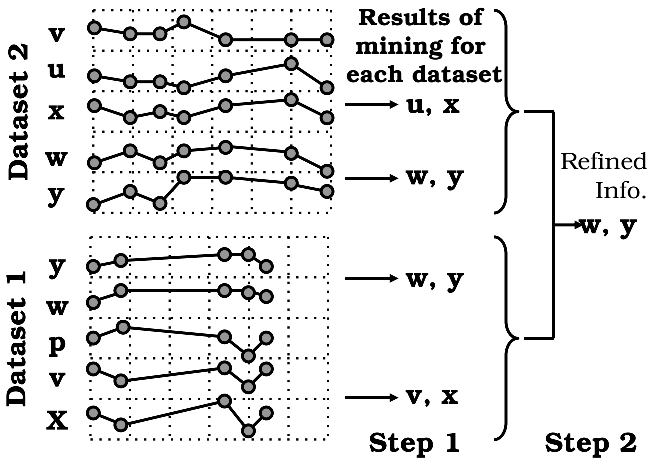

Figure 2: The two steps of information mining depicted on a hypothetical example run.

Case Study 1: Clinical Trials Data

A clinical trial [3] is a research study to answer specific questions about vaccines or new therapies. Clinical trials are used to determine whether new drugs or treatments are both safe and effective. In these trials, patients are assigned a treatment or a placebo and measurements for certain analytes (blood ingredients) are taken at intervals. These measurements can be represented as a time series for each analyte.

Hepatotoxicity (liver damage) is a common problem in drug treatment trials but is observed only indirectly through bio markers measured in the blood. This creates the need to infer an individual's unobserved liver function dynamically using blood tests and other patient baseline characteristics. Each phase of Clinical Trials is very expensive for the companies. If a severe side effect of a drug is not detected in clinical trials it costs significantly more to the company. We have developed methodologies that enable the estimation of the effect of drugs on hepatotoxicity, and thus determine early on whether new drugs should not be pursued into the next phase.

For step 1, we utilize K-Medoid [4] algorithm to cluster analytes. For each patient, the pairs of analytes that come together in the same cluster are saved as the output of the first step. The output can be considered as a transaction data (e.g., market basket data [5]) that includes information about which sets of analytes appear together. In other words, each cluster is mapped to a single transaction containing analytes.

For step 2, a frequent item set mining algorithm [5] is run to find strongly related analyte groups, most of which are shown to have high biological correlation. Frequent item sets are defined as the sets of items that co-occurred often enough to pass a given threshold, the support limit. We apply a *second level of pruning* and select a set of item sets using the confidence values of their corresponding association rules. Confidence for an association rule $X \Rightarrow Y$ is the ratio of the number of transactions that contain $X \quad Y$ to the number of transactions that contain X [5]. In our context, the confidence for an association between a set and its subset is described as (number of times members of set co-occurred in the same cluster)/ (number of times members of the subset co-occurred in the same cluster). So, while finding a group; not only the total number of co-occurrences of the group members is compared with a threshold (support limit), but also the ratio between this total to that of each one-item-less subset is compared with another threshold (confidence limit). A group is announced if it can pass both tests.

The result of the framework with the above setting is the frequently co-occurring clusters of analytes. As validated by our experts (Refs. [6] for more details), these clusters represent different groups of biological panels. The clusterings and the corresponding panels are given in the below table.

Group Name	Group Analytes
Transporter	Hemoglobin, Hematocrit, RBC Count
Acute Infection	WBC Count, Neutrophils(%), Neutrophils (abs)
Serum Protein	Total Protein, Albumin, Globulin, Calcium
Liver	SGOT(AST), SGOT(ALT), LDH

Appearing in the same cluster is the complement of not being in the same cluster. By the same token, if the algorithm can determine which analytes occurred in the same cluster more than the Support Limit, it can also determine which of them did not come together more than a new support limit(the difference between the number of patients (data sources) and the Support Limit). By using this approach for Step 2, a global panel of analytes that best represents the overall information in the data is effectively identified. It reduced the number of analytes from 43 to 8 (or less). One of the resulting global panels of size 8 is:

Hematocrit, Neutrophils(%), Total Bilirubin, Globulin, SGOT (AST), BUN, Creatinine, and Phosphorus.

This panel includes a member from each biological group in the data. A possible extension to the above is to identify a minimal set of analytes needed to monitor for a specific purpose (e.g., efficacy or safety). This would reduce the costs in clinical trials and both before and after market analyses of the drugs.

Case Study 2: Time-Series Gene Expression Data

As more laboratories acquire microarray technologies, a large amount of gene expression data becomes available for cross-validation information from different sources. Researchers can detect individual genes with similar behaviors across a varied set of environmental conditions or tissue types, or groups of genes that have similar patterns of behavior across some set of conditions or time points in a time-series. To achieve this goal, one needs to mine multiple gene expression data sets involving time-series with potential differences in lengths and interval sizes. The proposed framework can be applied to integrate results of multiple datasets as well as various mining techniques for the same data set.

Clustering has been so far the fundamental way of mining microarray data [7, 8, 9]. As in other data mining tasks, selecting the appropriate distance function to measure the similarities/dissimilarites between genes is essential. Each function has its own strengths and drawbacks, and is designed to capture a particular kind of relationship. It is clear that any single clustering method with any single distance metric would not be enough to capture all types of relationships.

Mining multiple gene expression data sets involves time-series with potential differences in lengths and interval sizes. The data is heterogeneous, and the sensitivity of the experiments varies with the source [10]. A global mining of data would cause inaccuracies even with an extensive preprocessing. We utilize the Information Mining Framework to find the coregulated genes of a queried gene in this heterogeneous data from various datasets and/or with respect to four distance functions: Correlation Coefficient [11], Longest Common Subsequence Similarity (LCSS) [12], Dynamic Time Warping(DTW) [13], and Qualitative [14].

For Step 1, we develop Strong Group Algorithm (SGA) where genes are grouped around the query gene and ranked into shells that correspond to different levels of similarity. It utilizes a similarity threshold to find all the members of the related gene group where each has a closer pairwise distance than the threshold to each other.

In a microarray dataset, genes are directly or indirectly related to each other since they represent inter-connected events in the cell. As a result genes profiles are expected to appear as closer points in value space. However, discriminating genes accurately, and not eliminating close points adequately is a challenge. A careless query can result in more than the desired number of genes in the result set. Hence, strong relationships has been explored to make this process worthy. A gene has many relationships. Hence, to preserve strong relationships, a gene is allowed to occur in more than one cluster.

In SGA, each gene is considered as the center of a cluster, and all genes within a similarity threshold, Υ, are pulled into this cluster. Each gene behaves as the owner of a cluster; hence at the beginning we have as many clusters as number of genes. Further, each cluster is pruned with respect to the same threshold, , Υ, separately as follows. First, the genes in a cluster are sorted according to their distances to the owner gene. Then one by one in ascending order of the distance, their membership to the group is reconsidered. For each gene in the cluster, if it is within the threshold of all unpruned members closer to the owner gene of the current group then it is not pruned, otherwise it is pruned. So at the end of this step we have as many clusters as number of genes. Clusters for two different genes can be equal to each other, hence, identical clusters are eliminated, keeping only one.

For Step 2, we implement set operations where the information mined with different metrics or from different sources are integrated. This makes the capture of specific relationships such as shifts, negative correlations, etc. feasible. For example, if the researcher is interested in the genes that are related with a phase difference (i.e. time-shift), these kind of relationships are best captured by LCSS, Qualitative, and DTW. So, the relationships can be captured by getting the difference from the intersection of the resulting sets of these three metrics to the resulting set of correlation coefficient.

We demonstrate our framework on 80 time-series datasets of yeast (Saccharomyces cerevisiae) from Stanford Microarray Database [15]. In order to evaluate the relationships within the groupings, we have utilized a recently produced pairwise relational list on yeast by Troyanskaya et al [16]. We have selected 127 genes with the highest reported biological relationships in that list and compared them through 80 time series at hand with 4 different metrics and 8 different thresholds. We generated 2560 different result sets. As these genes are expected to be close to each other, it is a challenge to return results that contain a low ratio of false positives (FPs), i.e. to return a set of related genes mixed with few unrelated genes. This challenge is addressed by our SGA algorithm. The recall on these results were increasing by the increase in the number of datasets analyzed and the number of metrics being used.

We combined the resulting sets from multiple datasets for a metric, and a similarity threshold value pair by finding the relationships appearing frequently in the 80 datasets. As a result of combining the sets for different datasets, we have a result set for each threshold value and metric pair. When results of four metrics were combined by the intersection operation, the precision of the results has improved from 0.67, which is the highest value achieved by a single metric (Correlation Coefficient), to 0.78 for a similarity threshold of 0.85. Figure 3 presents the effect of intersecting the metrics.

Figure 4 presents the effect of uniting the metrics for the same similarity threshold, i.e. 0.85. Within each dataset number of known relationships as well as unknown relationships existed and as we increased the number of metrics the number of relationships captured increased as well.

We have also compared the rate of already discovered relationships to the rate of false discoveries. Our analysis showed that when the metrics were used more restrictively the ratio between true(TDR[1]) and false(FDR [2]) discoveries were

[1]TDR: When a relationship was known we have considered it as a true discovery

[2]FDR: When a relationship was not known we have considered it as a false discovery

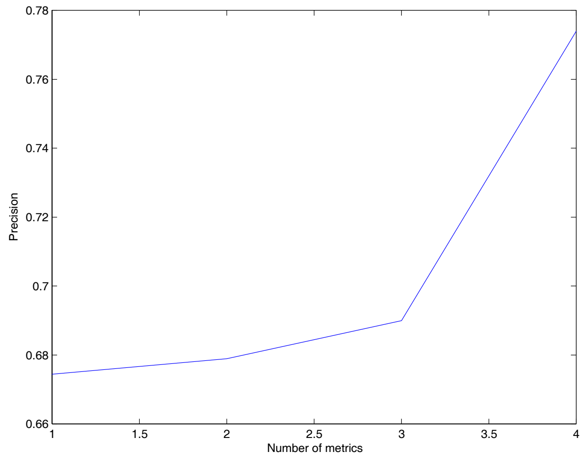

Figure 3: Increasing Number of Metrics "Intersection"

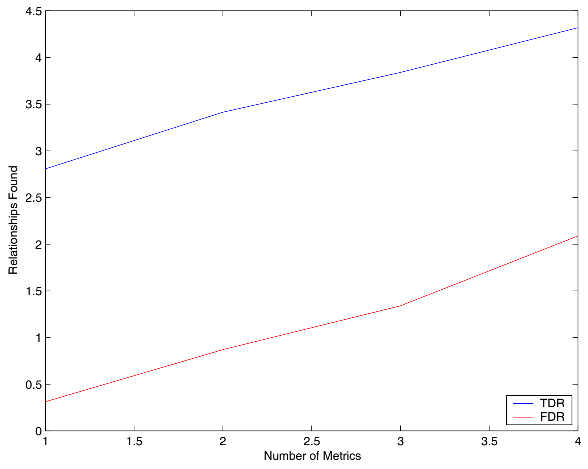

Figure 4: Increasing Number of Metrics "Union"

changing in favor of true discoveries. As Figure 5 indicates when a dataset is evaluated with higher thresholds the true discovery rate over false discovery rate increases.

Conclusion

We presented our framework, Information Mining, to extract consensus results from multiple sources, and showed its effectiveness in two important classes of biomedical data sets: Clinical Trials and Microarray data. Both applications share the commonality of having heterogeneous and high dimensional time sequences where the current data mining approaches fail to work. These applications share another similarity of having significant groups of homogeneous subsets. We proposed a two-step process to mine the data and refine the information to generate high quality results. The process is also applicable for merging results of various instances of a mining algorithm with different parameters, such as clustering with multiple metrics. The results on both datasets show that mining homogeneous subsets followed by integrating their results leads to improvement of the accuracy of the results.

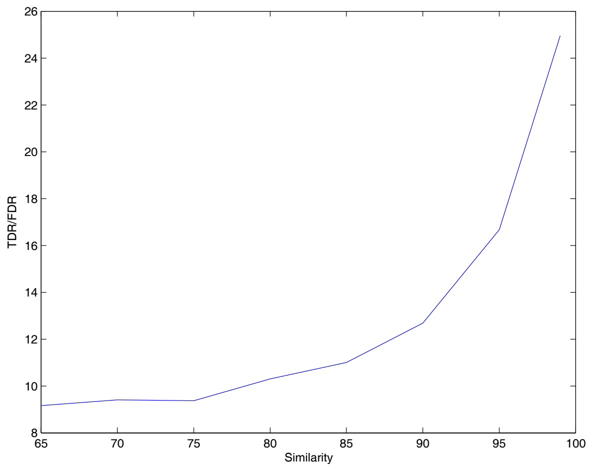

Figure 5: Increasing Similarity

Our results from Clinical Trials data have verified two biological panels (of blood analytes) already well-known in the medical field. Besides the well-known groups, we also have identified biological groups that are not commonly used. Our colleagues involved in this industry-sponsored project have already been using the results of the proposed technique. The statisticians used the global panel of analytes in developing accurate data distribution tests (such as multivariate normality) for high dimensional clinical trial data sets. The mathematicians involved in the project used the results in modelling the behavior of the human body in response to drug treatments. We expect several other uses of these results by researchers in pharmacology, biomedicine and biology.

We show the effectiveness of merging results from multiple metrics in addition to data from multiple sources, through applying our framework to Microarray data. It is clearly demonstrated that each metric has its own strengths and weaknesses which are clear when the metrics are applied individually. Combining results of them through set operations leads the investigators to answers of more specific questions with higher accuracy.

1. REFERENCES

[1] H. Ferhatosmanoglu, AS Tosun and A. Ramachandran, *Replicated Declustering of Spatial Data*,Proceedings of the Twenty-third ACM SIGMOD- SIGACT-SIGART Symposium on Principles of Database Systems (PODS), pp. 125–135, Paris, France, 2004.

[2] EM Knorr, RT Ng, *A unified notion of outliers: Properties and computation*, Proceedings of the Third International Conference Knowledge Discovery and Data Mining (KDD), pp. 219-222, 1997.

[3] National Library of Medicine, "*Information on Clinical Trials and Human Research Studies*", http://http://www.clinicaltrials.gov/.

[4] L. Kaufman and PJ Rousseeuw, *Finding groups in data.*, John Wiley, 1990.

[5] R. Agrawal, T. Imielinski, and A. Swami, *Mining association rules between sets of items in very large*

databases., Proceedings of the 1993 ACM SIGMOD international conference on Management of data, pp. 207–216, 1993.

[6] F. Altiparmak , H. Ferhatosmanoglu, S. Erdal , DC Trost, "*Information Mining Over Heterogeneous and High-Dimensional Time-Series Data in Clinical Trials Databases*", IEEE Transactions on Information Technology in Biomedicine (TITB), Volume: 10, Issue: 2, pp. 254- 263, April, 2006.

[7] M. Eisen, P. Spellman, P. Brown , and D. Botstein, "*Cluster analysis and display of genome-wide expression patterns*", Proceedings of the National Academy of Sciences USA, vol. 995(25), pp. 14863–14868, 1998.

[8] F. Gibbons and F. Roth , "*Judging the quality of gene expression-based clustering methods using gene annotation*", Genome Research, vol. 12(10), pp. 1574–81, October 2002.

[9] LF Wu, TR Hughes, AP Davierwala, MD Robinson, R. Stoughton, and SJ. Altschuler, "*Large-scale prediction of saccharomyces cerevisiae gene function using overlapping transcriptional clusters*", Nature Genetics, vol. 31, pp. 255–265, 2002.

[10] T. Critchlow and Z. Lacroix(Editors), Bioinformatics - Managing Scientific Data. Morgan Kaufmann, 2003.

[11] Walker MG, "*Introduction to Statistics for Bioinformatics, with Applications to Expression Microarrays*", CSB Tutorial sessions, 2003.

[12] S. Erdal, O. Ozturk, D. Armbruster, Ferhatosmanoglu H and Ray W, "*A time-series analysis of gene expression data,*" in IEEE International Symposium On Bioinformatics and Bioengineering (BIBE), Taichung, Taiwan, pp. 366–378, Mar. 2004.

[13] J. Aach and GM Church ,"*Aligning gene expression time series with time warping algorithms*", Bioinformatics, vol. 17(6), pp. 495–508, 2001.

[14] L. Todorovski, B. Cestnik and M. Kline, N. Lavrac, and S. Dzeroski, "*Qualitative clustering of short time series: a case study of firms reputation data*", ECML/PKDD'02 workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning, pages 141–149. Helsinki University Printing House, August 2002.

[15] PT Spellman, G. Sherlock, MQ Zhang, VR Iyer, K. Anders, MB Eisen, PO Brown, d. Botstein, and B. Futcher, "*Yeast Cell Cycle Analysis Project*", http://genome-www.stanford.edu/cellcycle/.

[16] OG Troyanskaya, K. Dolinski, AB Owen , RB Altman, and D. Botstein, "*A Bayesian framework for combining heterogeneous data sources for gene function prediction (in Saccharomyces cerevisiae)*", Proceedings of the National Academy of Sciences, vol. 100(14), pp 8348–8353, 2003.

Integrating Projects from Multiple Open Source Code Forges

Megan Conklin
Elon University
Box 2320, Elon, NC, USA 27244
(01)336-278-5204
mconklin@elon.edu

ABSTRACT

Much of the data about free, libre, and open source (FLOSS) software development comes from studies of code forges or code repositories used for managing projects. This paper presents a method for integrating data about open source projects by way of matching projects (entities) across multiple code forges. After a review of the relevant literature, a few of the methods are chosen and applied to the FLOSS domain, including a comparison of some simple scoring systems for pairwise project matches. Finally, the paper describes limitations of this approach and recommendations for future work.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics – *product metrics, complexity metrics.*

H.2.8 [**Database Management**]: Database Applications – *data mining.*

General Terms
Measurement, Algorithms.

Keywords
Open source software, data mining, software engineering data, entity matching.

1. INTRODUCTION

Free, libre or open source software (FLOSS) development teams often use centralized code forges, or repositories, to help manage their project code, to provide a place for users to find the product, and to organize the development team. Although many FLOSS projects host their own code repository and tools, many projects use the tools hosted at a third-party web site (such as Sourceforge, ObjectWeb, or Rubyforge). These code forges provide basic project/team management tools, as well as hosted space for the source code downloads, a version control system, bug tracking software, and email mailing lists. There are also directories of FLOSS software (such as Freshmeat and the Free Software Foundation directory) that try to gather into one convenient place material about projects interesting to a particular community.

Much open source software engineering research has been focused on gathering metrics from code repositories. Many aspects of the repository-based software development process have been studied in depth, and repository data collection is important for these studies (see [3] for background). The FLOSSmole project [6] was created to consolidate metadata and analyses from some of these repositories and directories into a centralized collaboratory for use by researchers in industry and academia. As of this writing, FLOSSmole includes data and analyses from Sourceforge, Freshmeat, Rubyforge, ObjectWeb, and the Free Software Foundation (FSF) directory of free software. One of the challenges mentioned in [3] in creating this kind of collaboratory is in integrating the data from these various sources. It seems reasonable that a project might be listed on several directories *and* have a listing on a code forge. However, sometimes a project will be listed in multiple forges too, usually because the project has migrated from one forge to another over time, or because the project wishes to "grab" the unique namespace for its project on a certain forge so it will register at that forge without an intention to ever actually use that space.

In any case, when integrating project data from multiple sources, we must first identify which project pairs are matches. In other words, we want to find out which projects are listed on multiple forges. For example, is the *octopus* project on ObjectWeb the same as the *octopus* project on Sourceforge or the project also called *octopus* on Freshmeat? If we can devise a scoring system for determining whether a project pair is a match, then can we automate the matching process?

The focus of this paper is entity matching (and duplicate identification) for this kind of data integration, as applied to the domain of FLOSS projects. Section 2 outlines some terminology from the study of data integration problems and gives a background of entity matching algorithms. Section 3 describes the FLOSS domain in terms of entities and duplicates. Section 4 gives an example of applying some of the algorithms for entity matching to this domain. Section 5 outlines limitations of this work and gives recommendations for future study.

2. ABOUT ENTITY MATCHING

The act of integrating multiple data sets and finding the resulting duplicate records ("matches") is nearly as old as database processing itself. In practice and in the literature, this set of processes is known by many names: merge/purge, object identification, object matching, object consolidation, record linkage, entity matching, entity resolution, reference reconciliation, duplicate identification, and name disambiguation. The term *entity matching* will be used in this paper.

Within the larger activity of data integration, the act of matching entities is not to be confused with the act of schema reconciliation. Schema reconciliation refers to the act of matching up columns or views in different data sources, and using data or metadata to make the match. For a trivial example, suppose a field in Table A is called *url* but it is called *home_page* in Table B. To resolve these schemas, the analyst could create a global schema or view that encapsulates both underlying schemas. This task can be done manually, or can be automated through various machine learning techniques [2,5,8]. Schema reconciliation and entity matching are related, but not identical, tasks of data integration. Most often the schema reconciliation will happen first, followed by the "merge" task, and finally by the eventual "purge" of duplicate data.

2.1 Agree/Disagree and Frequency-Based Matching

The simplest and oldest form of entity matching is the agree/disagree method: take two data sets A and B and compare them pairwise for matches based on one or more attributes. The pairs will either agree or disagree on zero or more of the attributes, and thus a weight for the match can be determined.

In trying to improve agree/disagree entity matching, [7] uses frequencies of values to determine the probability of a match (see [10] for another brief explanation of this work). Newcombe in [10] asserts two premises: (a) that matches are easier and more accurate when particular values in the fields are considered (as opposed to only considering all pairwise matches between two data sets on a common field), and (b) that two rare values are more easily and accurately matched than two common values. The example given in the Winkler paper is to compare the process of matching the following: two records listing the name Zbigniew Zabrinsky, two records listing the name James Smith, and any two records with first and last names. The two records for Zbigniew Zabrinsky are likely to be more easily and accurately matched than James Smith due to the rarity of the field values. The author also writes that, on a practical note, first and last name values alone are usually not very good for performing matches of human beings. He suggests first that other attributes such as address, date of birth, etc. should be considered in order to reduce false positives, and also that more sophisticated techniques for string comparison should be implemented.

2.2 Disjoint Sets

In [5], the authors consider the problem of how to match 'person' records using disjoint attributes and a 'typical person' profile. For instance, the example given in the paper is that the two records {Mike Smith, age 9} and {Mike Smith, salary $200,000} are not likely to be the same person based on a profile indicating that a typical person with an annual salary of $200k is older than 9 years. The authors compare their system to a traditional agree/disagree system of matching, and show that disjoint attributes can be effective if paired with shared attributes.

3. ENTITY MATCHING METHODS FOR FLOSS DATA

This portion of the paper describes the way each of these entity matching methods can be applied to integrate disparate sets of projects aggregated in the FLOSSmole collaboratory [6]. FLOSSmole is a research collaboratory for open source data. The project contains raw data and analyses from six of the available source code forges, and contains data from as far back as 2003. The FLOSSmole data is collected in two ways: through donations either given by people who run the forges or by researchers, and through automatic spidering of the code forge web sites. After the data is donated or collected, it is cleaned and aggregated. Finally the data is provided in various raw and summary formats for use by the general public under an open source license.

By way of introduction to the FLOSSmole data, Table 1 shows a partial list of the project attributes available for each of the repositories/forges in FLOSSmole at the time of this writing. These project attributes are the most likely candidates for the job of matching projects. (There are dozens of other attributes about each project in FLOSSmole, such as registration date or project status or number of downloads, but these are not likely to be helpful in matching projects across repositories.)

Table 1. Project metadata: relevant attributes for matching projects, (FLOSSmole, Dec, 2006)

Attribute	**Forge** *Sourceforge, Freshmeat, Rubyforge, Objectweb, Free Software Fndn.*				
	SF	FM	RF	OW	FSF
Short Name (unixname)	X	X	X	X	X
Long Name	X	X	X	X	X
Description	X	X	X	X	X
URL	X	X	X	X	X
License Type(s)	X	X	X	X	X
Programming Language(s)	X	X	X	X	X
Operating System(s)	X	X	X	X	
Topic(s)	X	X	X	X	
Intended Audience(s)	X	X	X	X	
User Interface(s)	X	X			X
Environment(s)			X	X	
Developer(s)	X	X	X	X	X

Most of these attributes shown in Table 1 are self-explanatory. However, some confusion can arise when differentiating between the short name and the long name for a project. The short name is usually an internal-to-the-repository name that is given to the project at the time of its creation. Some repositories use this as a sort of primary key for the project in its database. The long name of a project is the more descriptive name for a project. It can change over time, it can include spaces and special formatting characters, and it typically more descriptive than the short name. Values for all of the attributes shown in the list in Table 1 are chosen by the project administrators, and except for short name and long name, they can all be NULL. License type, operating system, topic, audience, interface, and environment can have multiple values.

The next three sections describe a few of the obvious choices for attributes from this list that can be used to establish matches between projects. One choice from Table 1 that may initially look promising is "List of Developers". Since this attribute is actually a list of developers who work on each project, what better way to differentiate or match two projects? (If the list of developers is the same for the two projects, then the two are a match.) The problem with this is that developers are entities themselves, and matching developers between repositories requires an entirely separate list of attributes (developer name, developer email, developer skills, role on project, etc). In addition, developer information is often intentionally obfuscated by the developer themselves, and in some cases by the forge maintainers. Finally, as the authors in [9] described, there could be significant privacy implications to using developer data without the express consent of the developers themselves.

Section 5 discusses broadening project entity matching to include developers, but the remainder of this paper will exclude developers as entities and will retain the focus on project matching only.

3.1 Matching by URLs

Matching projects by URL has two possible: projects listed on different forges might both display the same external URL, or projects on one forge might actually list the project site on a competing forge as the home page of record. The diagram shown in Figure 1 depicts each forge/directory in FLOSSmole and how many of its projects list another forge as the actual hosting home page. For example, in the diagram, the topmost arrow shows 11 projects on the FSF that actually have Rubyforge listed as the home page. The arrow notation is used to show a direction of the relationship (e.g. 10,044 Freshmeat projects show a home page on Sourceforge, but only 4 Sourceforge projects list a Freshmeat home page). Pairs of forges with no URLs in common are not shown. (No Rubyforge projects list ObjectWeb URLs, and vice versa. Also, as is befitting its status as a directory and not a repository, the FSF directory is not listed as the home page of any projects from the other repositories, so these empty relationships are not shown in the diagram.)

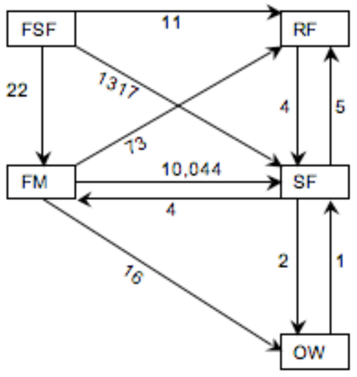

Figure 1: Number of projects at each repository that list a home page at another repository

3.2 Matching by Project Names

Figure 2 shows the number of short project names shared in common between each pair of projects. For instance, *starfish* is a project listed on both Sourceforge and Rubyforge. On Rubyforge, it is described as a "tool to make programming ridiculously easy", but on Sourceforge the *starfish* project is described as a password management application. There are 470 projects with shared names on Rubyforge and Sourceforge. A similar problem exists between the project names on Sourceforge and ObjectWeb. For example, the project called *octopus* exists on both these forges and appears to be a completely different application: on Sourceforge this is an Eclipse plug-in, but on ObjectWeb *octopus* is an ETL data warehousing tool. Of the 125 applications (total) listed on ObjectWeb, 41 have names that are shared with a Sourceforge project. The Sourceforge project may (as in the case of *lemonldap*) or may not (as in the case of *octopus*) be the same project. On Freshmeat, there also is a project called *octopus*, but this one is a financial trading application.

Most forges require projects to have a unique name (sometimes called the "unixname") within that forge. For example, once a project called *starfish* has been added to Sourceforge, another one cannot be added with the same short unixname. However, multiple projects can have the same "display name"; Sourceforge projects *starfish* and *xstarfish* both have the display name of "starfish". On Sourceforge, 44,112 (39%) of projects have unixnames that are different from their display names (December 2006 FLOSSmole data). Note that the FSF directory has only a requirement for case-sensitive uniqueness in project names. The FSF lists project pages for both *ANT* (telephony application) and *ant* (build tool). There are 54 such ambiguously named projects listed on FSF.

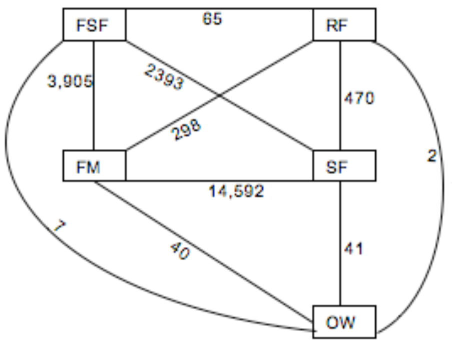

Figure 2: Number of projects at each repository that share an identical short project name

3.3 Matching by Other Attributes

It may be possible to determine the accuracy of each matched pair further by attempting to match the project owner or developer names, emails, or usernames as in [9]. Or, it may be possible to find a matched pair through the textual description of the project, or through the project license type, the programming language(s), operating system(s), or other metadata about the project. Each of these possible match fields requires that the project administrator has accurately filled in the metadata for his/her project. If the administrator never bothered to fill in the programming language for the project on one or both of the sites where the project is listed, then it will not be possible to disambiguate by finding a match on this item.

Table 2 summarizes a few of the most common attribute statistics for Sourceforge projects. It is also interesting to discover that of those 74% of projects that list a license type, over half use the Gnu General Public License (GPL).

Table 2: Numbers of Sourceforge projects with and without certain attribute data, (FLOSSmole, Dec, 2006)

Project Attribute	Projects listing at least one	Projects listing none
Programming Language	82,969 (73%)	29,946 (27%)
License Type	84,102 (74%)	28,813 (26%)
Operating System	78,334 (69%)	34,581 (31%)

3.4 Advanced Methods

In our attempt to match FLOSSmole projects by common URLs, names, or any other combination of attributes, we are still performing basic agree/disagree entity matching. Our brief review of the database literature on entity matching indicates that these methods do work for some cases, but can be optimized and improved. This section discusses the improvements we added, and Section 4 will explain the results of our application.

3.4.1 Frequency-Based Matching and String Metrics

The first improvement made to the agree/disagree entity matching is to consider how to apply a form of frequency matching on the name field. Recall that [10] explains that rare names (Zbigniew Zabrinsky) are more easily matched than common names (James Smith). "Rare" and "common" are determined by an already-existing set of names and their general frequency rankings in the population. In the case of FLOSS projects, there is no such ranking for software project names, but a corollary might be that projects with dictionary words for names (e.g. the *octopus* and *starfish* examples) are more likely to be non-matches than projects with unusual, non-dictionary names (e.g. *sqlite-ruby* or *lemonldap*). Because there is also a difference between the unique unixname and the non-unique display name for each project, we ask: which of these fields should be used to consider the frequency match? In Section 4, we answer with "both", and we experiment with scoring these matches differently.

Another improvement we considered was to look at string metrics for determining whether two projects were matches. Regular expressions can be used to compare strings in a simple fashion, while Hamming distances and Levenshtein distances [1] are more complex algorithms used to compare the similiarity of two strings. With Levenshtein distances, the higher the calculated distance value, the less similar the strings are. Distance is calculated based on the number of deleted, inserted, or swapped characters it takes to turn one string into another.

3.4.2 Disjoint Sets

The next improvement we considered was to use the notion of a disjoint set, as in 2.2. by listing which attribute values would likely never coexist. Initial ideas included the following possible disjoint sets: {op_sys=linux, prog_lang=asp}, {date_regist<2001, prog_lang=C#}. Not only are these rules fairly weak insofar as there are plenty of examples of projects that would violate them for various reasons, but unlike the age/salary information in the example case in 2.2, the number of records in FLOSSmole which match these disjoint sets is likely to be quite small. We conclude that in the FLOSS domain, it is more likely to be the case that matches can be found through simpler methods than disjoint sets. This is due to three factors: the low number of valid disjoint set rules we would be able to construct, the difficulty of applying disjoint set rules to our data when so many of the pairs are missing metadata on which these disjoint sets would be based, and the low number of duplicates that would not be identified by other, simpler methods.

Table 3: Trial One Sample Scoring Table

	Modifier
Home Page URLs match	+3.00
Short names match	+2.00
--if yes, is short name in the dictionary	-1.00
--if not, does Partial Name match?	+0.50
-- if partial name matches, is partial name in dictionary?	-0.25
Textual descriptions tokens match, per token match	+0.10
Long names match	+0.50
Programming language matches, per token match	+0.50
License matches, per token match	+0.50
Other project metadata matches, per token match	+0.50

4. APPLICATION

To apply entity matching methods to project data in FLOSSmole, we assumed a set of heuristics and associated weights for calculating whether the items in a pair are a match (Table 3). In Trial One, match modifiers were initially based on an intuitive sense of which matching criteria were important. In Trials Two, Three, and Four, we attempted various alternative scoring systems, and there is a description of these in Section 4.1.2. We are keenly aware of the limitations of this method; in the future (see also Section 5), we may wish to expand our methodology a bit by using statistical or information theoretical models for a more effective scoring system.

Table 4: Sample scores for matching project pairs, Trial One

Pair ID	Project Name	Src.A	Project Name	Src.B	Score
1	phpmyadmin	SF	8001 (phpmyadmin)	FM	6.9
2	octopus	SF	octopus	OW	1.0
3	octopus-ge	SF	octopus	OW	2.6
4	16120 (octopus)	FM	octopus	OW	1.5
5	13902 (ant)	FM	152 (ant)	FSF	4.1
6	sqlite-ruby	SF	sqlite-ruby	RF	6.9

4.1.1 Example

A short example of a table designed to hold the FLOSSmole pairs with their matching scores across multiple repositories might look like Table 4. Higher scores mean the pair is more likely to be a match, but it will be up to an individual analyst to decide where to "draw the line" for what score indicates a match. The highest score is around 8; the lowest score is 0. As shown in the table, the highest score could be higher if more attributes were added. Attributes included are programming language, operating system, and license because these are the fields whose values were most available and easiest to standardize over a variety of repositories.

(Table 1 showed that attributes like "environment", "interface", or "topic" are harder to standardize.)

Pair 1 shows Sourceforge project phpmyadmin matching an identically-named Freshmeat project. These projects share a short name (with low frequency count when compared to a dictionary word: +2), long name (+.5), URL (+3), and license type (+.5). Several key tokens are the same in each description (+.9: The matching tokens are 'MySQL', 'PHP', 'Web', 'administration,' 'alter', 'drop', 'database', 'delete', 'SQL').

Pair 2 shows Sourceforge project *octopus* with ObjectWeb project *octopus*. The short project names match (+2), but urls are different. The long project names are also different ('Octopus' and 'Enhydra Octopus'). Additionally, because the Sourceforge project *octopus* does not list any project metadata, it can't be matched very well with the ObjectWeb project of the same name using these additional attributes. Finally, these two entities share the dictionary word name 'octopus' (-1).

Pair 3 shows the project *octopus-ge* on Sourceforge and project *octopus* on ObjectWeb. These projects share a beginning partial string match, *octopus** (+1), but that string is a dictionary word (-.5). They share one programming languages (+.5), a license type (+.5), and one operating system (+.5). The textual description of the projects increases the score, since both use the token strings 'Enhydra Octopus', 'extraction', 'transformation', 'load*', 'ETL', and 'XML' (+.6). However, a closer read of the textual description field by a human being reveals that the Sourceforge project is actually a graphical editor for the ObjectWeb project. They are related projects, but not the same project. The combination of no URL score and low scores for the textual matches has (accurately) kept this project from a high score.

Pair 4 shows the attempted match between that same *octopus* project at ObjectWeb but now paired with the *octopus* project at Freshmeat. The projects have the same short name (+2 for similarity, -1 for dictionary), but different URLs, totally different textual descriptions, and share only the license type in common (GPL, +.5). Indeed, manual checking of this result shows that these two projects are not related.

Pair 5 shows the Freshmeat project *ant* matching with the Free Software Foundation project *ant* as follows: short name (+2), url (+3). However, the display names for this project are different ('ant' on FSF and 'apache ant' on FM). In addition, the common dictionary name 'ant' lowers the score somewhat (-1). Note that while there is only one significant matching token in the textual description (the word "Java", +.1), the entire first sentence of the two projects is identical. This indicates a strong need to refactor the scoring algorithm for textual descriptions.

Pair 6 shows that *SQLite-ruby* projects listed nearly identical information on both Sourceforge and Rubyforge. They share: the project home page (+3), short name (+2), the display name (+.5), one programming language (+.5), the operating system (+.5), and 4 significant text tokens (+.4), yielding a total score of 6.9.

4.1.2 Comparison and Evaluation

The problems with the Trial One scoring system seem obvious: first, calculating probably-insignificant token matching on completely unrelated projects takes up an enormous amount of processing time, and second, there is no "memory" for matching likelihoods based on the calculations that have already taken place. Trials Two, Three, and Four attempt to solve some of these problems. Tables 5 and 6 show the scoring systems used for these additional trials.

Table 5: Trial Two Sample Scoring Table

	Modifier
Home Page URLs match	+3.00
Short names match	+2.00
--if yes, is short name in the dictionary?	-1.00
Long names match	+0.50
Programming language matches, per token match, but only if score > 0	+0.50
License matches, per token match, but only if score > 0	+0.50

In Trial Two, we made two changes. First, we simply reduced the number of small-scoring attributes, while leaving the most decisive attributes in the scoring system. This was decidedly faster for processing, and eliminated some of the irrelevant "false positives" generated by cumulative low scores. We also introduced the notion of phased scoring by major and minor attributes. The remaining token match attributes (programming languages and licenses) were only scored *if* the project pair already had some sort of positive correlation from one of the other three major attributes (URL, short name, or long name). This seemed to solve the problem of trivial token matching, and solved some of the problems of processing time.

In Trial Three, we decided to explore solutions for the speed-of-processing issues caused by our massive Cartesian product generation (all projects in one forge times all projects in another forge). To do this, we considered what we know about how disjoint sets work (Section 2.2 and 3.4.2). We decided to invert our program logic a bit: instead of building up a huge list of projects and comparing all projects in forge A against all projects in forge B, we instead attempt to identify likely sets immediately (or eliminate disjoint sets immediately) and thus we can *discount* projects that are unlikely to be related. This will serve to reduce the size of the sets being compared. For example, we built a list of projects that used Java on forge A and another list of projects that use Java on forge B, then only compare these projects along the other attributes. This method didn't actually reduce the number of comparisons, since every project had one or more programming languages, but it did serve to reduce the size of the in-memory hashes being built.

In Trial Four we wanted to improve the accuracy of our partial name matches. We decided to use Levenshtein distances to measure the similarity between two tokens, such as between two short names. (In Trial One we had used simple regular expression-based partial string matches, so for example, *starfish* and *xstarfish* would match but *andychat* and *andy**s**chat**45*** would not match.) Levenshtein distances are calculated for each string pair based on the smallest number of characters that must be deleted, replaced, or inserted to make one string into another string. In our case, we choose to set an arbitrary threshold for similarity at 25%. The Levenshtein distance between *andychat* and *andyschat45* is 3 (or 25% of the longest string), which is (barely) enough to warrant token matching in programming languages and license types according to our threshold of 25% or less.

Table 6: Trial Four Sample Scoring Table

	Modifier
Top of chart same as for Trial Two: URLs, Names…	
If score = 0, measure Levenshtein distance between names, giving 0.5 if <25% difference	+0.25
Programming language matches, per token match, but only if score > 0	+0.50
License matches, per token match, but only if score > 0	+0.50

5. LIMITATIONS AND FUTURE WORK

Based on the experiments described in Section 4, entity matching is an interesting exercise, but is certainly problematic. One of the most obvious problems is that the scoring modifiers given in Tables 3, 5, and 6 are completely arbitrary and based on trial and error and an intuitive sense of the data. In the case of Trial One, there is a distinct possibility that a pair of projects could achieve a score of 4.0 by having a partial non-dictionary name match (+.5), five attributes in common (+2.5), and a handful of well-chosen tokens in the textual description (+.5), and yet these projects could be completely unrelated. Tweaking the scores to reduce false positives and false negatives in this way is tedious and inefficient.

In the subsequent trials, we attempted to reduce processing time and false positives/false negatives, but there are many other routes we could have taken to accomplish this. For instance, in the case of token matching, we look at the case of *ant*, for which there were very few singularly meaningful tokens in the textual descriptions, but the description as a whole matched perfectly. The use of dictionary word definitions for frequency matching may need to be refactored also. The *ant* match lost points because of this. We also recognize that open source developers and projects are a decidedly global population, and more languages than just English are used, so perhaps English dictionary matching is an arbitrary solution. Would other dictionaries be effective? Should non-dictionary strings that are also common in software development ("lib", "db", "php") be added to the dictionary?

Next, what about multi-way matches? We have given little attention to the problem (as presented in [6]) of how to merge multiple confidence scores after they've been created. Consider a project such as *sqlite-ruby* that appears on Sourceforge, Rubyforge, Freshmeat, and the FSF directory. What is the appropriate way to integrate its multiple scores? *Sqlite-ruby* is likely to have high scores on all 6 pair combinations, so a simple average might work, but what about a project like *ant* whose scores may vary more?

Section 3 mentioned the possibility of matching projects based on the lists of developers on each project. Before doing this, it would be necessary to use similar entity matching methods to actually match developer entities as well. As is described in [9], matching developers also leads to a few additional complexities: "real" emails are most often not available for public lists of developers on code repositories, name matching with developers could be even more complex than matching on names for projects because of similarities in names and spellings, and of course, developer privacy is always a concern when integrating personal data points.

One final recommendation for future work is to remember some of the work being done on sites like Krugle, Swik, and the Galactic Project Registry to standardize the notion of a project name. Krugle is a source code search engine that actually uses some FLOSSmole data to populate its list of projects. Swik is a wiki of information about individual open source projects; it gets some of its initial information from FLOSSmole as well. The Galactic Project Registry is attempting to put together a plan for being "the One True Known Up-To-Date Source" for project names and DOAP (description of a project) information on each project. Each of these projects probably would benefit from this work in entity matching and duplicate identification across repositories, and perhaps they can contribute to the conversation about the best way to achieve this goal.

6. REFERENCES

[1] Algorithms and Theory of Computation Handbook, CRC Press LLC, 1999, "Levenshtein distance", in *Dictionary of Algorithms and Data Structures* [online], Paul E. Black, ed., U.S. National Institute of Standards and Technology. 11 June 2007. (Accessed 15 June 2007) Available from: http://www.nist.gov/dads/HTML/Levenshtein.html

[2] Batini, C., Lenzerini, M., Navathe, S. A comparative analysis of methodologies for database schema integration. *ACM Comp. Surveys*, 18:4 (1986). 323-364.

[3] Conklin, M. Beyond low-hanging fruit: seeking the next generation of FLOSS data mining. In *Proc. 2nd Intl. Conf. on Open Source Systems.* (Como, Italy, June 2006). Springer, New York, NY, 2006. 47-56.

[4] Doan, A., Domingos, O., Halevy, A. Reconciling schemas of disparate data sources: A machine learning approach. In *Proc. of the SIGMOD conference.* (Santa Barbara, CA, USA, 2001). ACM Press, New York, NY, 2001, 509-520.

[5] Doan, A., Lu, Y. Lee, Y., Han, J. Object matching for information integration: A profiler-based approach. In *Proc. of the IJCAI Workshop on Information Integration and the Web.* (Acapulco, Mexico, 2003). 53-58.

[6] Howison, J., Conklin, M., Crowston, K. OSSmole: A collaborative repository for FLOSS research data and analyses. In *Proc. 1st Intl. Conf. on Open Source Systems.* (Genova, Italy, June 2005). 54-59.

[7] On, B-W., Lee, D., Kang, J., Mitra, P. Comparative study of name disambiguation problem using a scalable blocking-based framework. In *Proc. of 5th ACM/IEEE-CS Joint Conf. on Digital Libraries.* (Denver, CO, USA, 2005). 344-353.

[8] Rahm, E. and Bernstein, P. A survey of approaches to automatic schema matching. *VLDB J:10*, 2001. 334-350.

[9] Robles, G. and Gonzalez-Barahona, J. Developer identification methods for integrated data from various sources. In *Proc. of Mining Software Repositories Workshop (MSR 2005)* (St. Louis, MO, USA, 2005). 1-5.

[10] Winkler, W. *The State of Record Linkage and Current Research Problems.* Technical Report, Statistical Research Division, US Bureau of the Census, 1999.