Relational Ensemble Classification

Christine Preisach
Computer-based New Media Group (CGNM)
Department of Computer Science
University of Freiburg, Germany
preisach@informatik.uni-freiburg.de

Lars Schmidt-Thieme
Computer-based New Media Group (CGNM)
Department of Computer Science
University of Freiburg, Germany
lst@informatik.uni-freiburg.de

Abstract

Relational classification aims at including relations among entities, for example taking relations between documents such as a common author or citations into account. However, considering more than one relation can further improve classification accuracy.

In this paper we introduce a new approach to make use of several relations as well as both relations and attributes for classification using ensemble methods. To accomplish this, we present a generic relational ensemble model, that can use different relational and local classifiers as components. Furthermore, we discuss solutions for several problems concerning relational data such as heterogeneity, sparsity, and multiple relations. Finally, we provide empirical evidence, that our relational ensemble methods outperform existing relational classification methods, even rather complex models such as relational probability trees (RPTs), relational dependency networks (RDNs) and relational Bayesian classifiers (RBCs).

1. Introduction

There are many possibilities for saving, handling and publishing a large amount of data nowadays. This leads to problems related with the retrieval of information in a convenient way. In order to facilitate this retrieval, we need to structure and organize this information. One way to do this is to classify the information in particular categories. Usually this is done by considering only local features like text of entities. In text classification, it has been assumed that entities are independent, however in most cases this assumption does not hold because of the existing relationships among the documents. Most documents, especially scientific publications, are related through explicit links like citations or in an implicit manner, for example, if they are written by the same author or published in the same journal. These relations could be exploited using *relational classification* or *relational learning* to improve classification accuracy. Here the classes of related entities are used as predictor variables. Recently, especially relational methods, which use *collective classification* (also called *collective inference*) [11], i.e. iterative algorithms, which exploit the relational autocorrelation of variables of connected entities, received attention.

But in general, there is not just one relation to consider in *relational classification*, but several. In literature, multiple relations are combined either by using a set of neighbor class predictor variables per relation in a single local classifier [20, 22] or by merging the relations and summing the weights of common links [16]. Here, we will present a new approach of combining different relations using *ensemble classification* [5].

Often there are not just relations, but also attributes of the objects under consideration, such as the textual content (e.g., represented as a bag-of-words) for document objects. If such local attributes are to be considered, in literature typically this is accomplished by adding them as predictor variables to the relational classifiers. We, in contrast, combine local attributes with relations using again *ensemble classification* methods.

Recently *ensemble classification* aroused interest in the area of Machine Learning [5, 26, 2] due to its ability to reduce variance and bias and consequentially increasing accuracy. However, on our best knowledge it has not been used for the aim of combining several relations.

In this paper we will make the following main contributions:

1. Formulate the *relational classification* problem and describe how several problems such as heterogeneity, sparsity and multiple relations concerning relational data can be handled.

2. Introduce our generic *relational ensemble model* that consists of a local, a *relational* and an *ensemble classification* component.

3. Present some *relational classification* methods that use *collective classification* and introduce two new simple relational methods.

4. Evaluate the presented *relational classification* methods using *ensemble classification* on two scientific publications datasets, one of them being very complex. Compare the achieved results to well known complex *relational learning* algorithms like RBC, RPT and RDN as well as to a text classifier.

2. Related Work

One of the earliest approaches considering relations between entities was by Chakrabarti et al. [3], who proposed a probabilistic model for classification of web pages using the content of the classified page, the class labels of linked pages as well as the content of these linked pages. They also used a *collective classification* algorithm (relaxation labeling) and showed, that using the content of a web page and the class labels of linked web pages increases predictive accuracy. The algorithm "Hubs and Authorities" also considers the links to other web pages and [13] calculates the weights (hubs and authorities) iteratively.

More recently, researches have been done for instance on relational probability trees [20], which is a complex learning algorithm taking into account relations among entities and using probability trees [23]. Furthermore on relational dependency networks [22], a type of probabilistic relational model and on regression models using link distributions [15], as well as on simple models like probabilistic relational neighbor (PRN) classifier [16] or other simple methods in [1]. Although most of them use several relations, these relational methods did not apply *ensemble classification*, neither for the combination of multiple relation, nor in order to combine a local classifier with a relational classifier. Moreover, they either are not considering local features or if doing so, they incorporated them additionally as predictor variables to the relational classifier. Macskassy and Provost [16] too, considered multiple relations, they represented the considered relations in a single graph (merging the relations and summing the weights of the common links) and performed their algorithms only once on this representation.

Fürnkranz [7] used *ensemble classification* in the area of hyperlink classification. *Ensemble classification* methods have been incorporated in order to combine the results of the predictions for each hyperlink of a target page. He showed, that this technique performed as well as a classifier considering only the content of the target page or even better. In contrast to his approach, we do not combine the prediction for each instance of a relation (link) but only for each relation type. Heß and Kushmerick [8] described a different approach, they extended the iterative algorithm of Neville and Jensen [21] and used voting to combine intrinsic and extrinsic features of web services. They showed, that with voting, one can achieve better results than with a text classifier or a relational classifier, that only appends the intrinsic features to the extrinsic features. This is similar to our approach of combining the results of a local and a relational classifier; however, besides *voting*, we additionally used a more powerful *ensemble classification* method, namely *stacking* and combined multiple relations by using *ensemble classification*, furthermore we have evaluated our methods on a public data set.

3. Problem Formulation

Before we formulate the problem of *relational classification* considering several relations, we define the problem of traditional classification. In traditional classification objects are described as vectors of values of some attributes (or variables) A. As we will need to distinguish between objects with eventually identical attributes, we will describe objects by an ID $x \in X$ and their attributes by a map $a : X \to A$, i.e., $a(x)$ corresponds to the usual description of an object by its attributes.

For a traditional classification task of objects into possible classes C, a set $X_{\text{tr}} \subseteq X$ of objects together with their classes $c : X_{\text{tr}} \to C$ is given for training and another set $X_{\text{tst}} \subseteq X$ of objects together with their classes $c : X_{\text{tst}} \to C$ is given for evaluation (with $X_{\text{tr}} \cap X_{\text{tst}} = \emptyset$). The task then is to learn an attribute-based classifier model

$$\hat{c} : A \to C \tag{1}$$

s.t. $\hat{c}(a(x))$ equals $c(x)$ for as many $x \in X_{\text{tst}}$ as possible, e.g., the misclassification rate

$$\text{err}_{X_{\text{tst}}}(\hat{c} \circ a, c) := \frac{|\{x \in X_{\text{tst}} \,|\, \hat{c}(a(x)) \neq c(x)\}|}{|X_{\text{tst}}|}$$

is minimal.

In *relational classification* there is, additionally to the attribute-value data for objects, relational data about the objects. In the simplest case, there is a single binary relation $R \subseteq X \times X$ between objects under consideration. Classifiers now can make use of additional information about related objects, e.g., their attributes or their classes, if they are known. As some of the related objects may themselves belong to the testset, *collective classification* can be used to derive consistent predictions. To make use of relational information for classification, it must be propositionalized, i.e., converted to suitable attributes. One of the simplest propositionalization is to calculate class frequencies of related classes, i.e.,

$$\text{freq}_c(x) := \frac{|\{x' \in R_x \,|\, c(x') = c\}|}{|R_x|}, \quad x \in X, c \in C$$

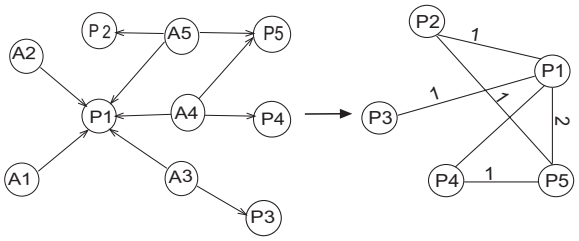

Figure 1. Heterogenous and Homogenous Representation

with $R_x := \{x' \in X \,|\, (x, x') \in R, c(x) \neq .\}$ the set of all objects related to x with known classes (a dot denotes a missing value). We will see more complex propositionalization schemes in section 6.2.

In general, there can be more than one relation, relations can hold between more than two objects, and relations can involve further sets of objects, e.g., for classifying books, also authors and journals could be important related objects of a different type. In this case, a binary homogeneous relation must be derived to apply most *relational classification* methods.

4. Relation Engineering

Here we discuss, how relational data can be represented, how multiple relations can be combined and how to handle sparsity.

First of all, we present possible views of relational data and its representation. Relational data is usually heterogenous but it can be viewed also in a homogeneous way.

Heterogeneity with respect to relational data means, there exist more than one type of objects, namely a set of target objects $x \in X$ and one or more sets of other object types O. Instances of these object types can be related, so that one can have different types of relations $R_1 \subseteq X \times O_1$, $R_2 \subseteq X \times O_2$ etc. This heterogeneous view can be represented by a directed acyclic graph.

We have used the homogeneous view, similar to Macskassy and Provost [16, 17] in order to allow the usage of normal classification methods. In the homogeneous view we have only one object type, such that, there exist a set of target objects $x \in X$ and relations $R \subseteq X \times X$ between these objects. Of course, on a semantic level, we have different relations.

In the homogeneous view, relational data is represented by an undirected, weighted Graph $G(X, E)$ with a set of nodes X and a set of edges $E \subseteq X \times X$. The edges are annotated with a weight w, which is $w : X \times O \rightarrow \mathbb{N}$, while O can be equivalent to X. So that the objects O are incorporated in the weight of the edges. A higher weight indicates, that the relation between two nodes is more important. The weight of an edge is equivalent to the number of objects $o \in O$, which two target objects $x, x' \in X$ have in common.

In the case of scientific papers, a paper can be connected to another paper for instance, because they have been written by the same author or since they cite each other. In figure 1, one can see an example graph from the domain of scientific publications that shows the transformation process from a heterogeneous to homogeneous representation. On the left side, a heterogeneous graph is shown, where two node types exist, the author nodes (A) and the publication nodes (P), the links indicate the relation between an author and a publication. On the right side the corresponding homogeneous view of the graph is depicted, it contains only the publication node type, the edges display the relation *sameAuthor* and the weights mark the number of authors, that two papers have in common.

Another possibility to homogenize heterogeneous relational data is to use similarity measures. That means, the weight of an edge between two objects X is equal to the correlation of these objects. The correlation could be for instance calculated by similarity measures like Pearson Correlation or Cosine Similarity, which are often used in the field of Recommender Systems. This approach is not used in this paper, however we will further investigate this in future research.

Another problem concerning relational data is, how to combine several relations. There are different possibilities to do this. One is to unify the relations, which need to be combined, so that the new relation is $S = \bigcup_i R_i$ and the weights of the edges of the new graph are the sum of the weights of the edges in the original graphs. This approach is used by Macskassy and Provost [16]. We in contrast, consider each relation R_i individually and build a graph for each relation. Then perform *relational classification* like described in the "Problem Formulation" section and combine the results with *ensemble classification* methods, which will be described in section 7. The main difference of these approaches is, that in the first approach the relations are combined on the data level and in our approach they are combined on the result level.

The next problem that we address in this paper is the sparsity of relations. It could happen, that some instances have only very few or even no neighbors. The problem of having no neighbors we do not address here, such instances are simply assigned with the prior class or the class proposed by a local classifier. We will only address the problem of few neighbors. To cope with this sparsity we consider not only the direct neighbors of an instance but also neighbors, which are located some hops away from the target instance. In order to avoid noise by using irrelevant neighbors we restrict this approach to the neighbors, which are located on

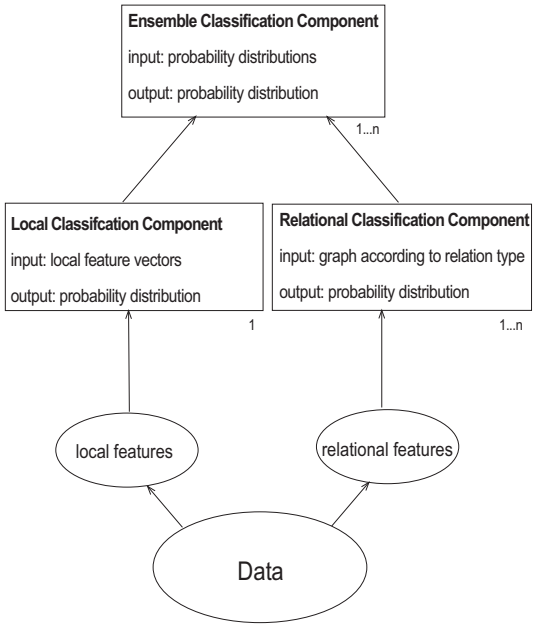

Figure 2. Relational Ensemble Model

a path of length two and even more important, only if instances are sparse with respect to the number of neighbors, additional neighbors will be considered. We call this approach *PRN2Hop*, it is introduced in section 6.1 more precisely.

5. Relational Ensemble Model

In this section we describe our generic *relational ensemble model*. It is composed of three main components, the *relational*, the local and the *ensemble classification* component (see fig. 2). The *relational ensemble classification process* is listed in algorithm 1. It describes a classification process, which uses all components of the model.

The individual components are:

Relational Classification Component For the *relational classification* component, we use the graph representation depicted in the right part of figure 1. For each relation (e.g. *sameAuthor*, *sameJournal* or *citation*) we build a new graph and apply *relational classification* methods (to be introduced in section 6) using *collective classification*. The input of the relational classification component is a set of instances containing relational features. The output of a *relational classification* algorithm is a probability distribution over the target variable.

Local Classification Component The local classification component uses only local features. In the domain of scientific publications, the local classifier performs traditional text classification. The data is represented in the manner of *bag-of-words*, i.e. each document is represented by a feature vector containing an entry for each word, the value of an entry is the weight (e.g. *TFIDF*) of a word. Then a machine learning algorithm is performed, the output is a probability distribution over the target variable.

Ensemble Classification Component The task of the *ensemble classification* component is to combine several classifiers in order to increase the classification accuracy. The input of the component is a set of probability distributions over the target variable, whereas they stem from the local or relational components. Either a simple *ensemble classification* algorithm can be used, for instance *voting*, or a meta classifier, which learns a meta model of the probability distributions provided by the base classifiers. The output of the *ensemble classification* component is a probability distribution too. In our experiments we have used the *ensemble classification* component twice, first for the combination of the results of the *relational classification* component and secondly as the last step in the model, in order to combine local and *relational classification*, as it is listed in algorithm 1.

6. Relational Classification Methods

Here we present both, simple and learning *relational classification* methods, all of them using *collective classification*.

Collective classification (also called collective inference) methods are iterative procedures, which classify related instances simultaneously. *Collective classification* exploits *relational autocorrelation*. While *relational autocorrelation* is an important property of relational data in which

Algorithm 1 Relational Ensemble Process

1. Classify instances using only local features
2. for each relation r:1 to m
   (a) Build a graph
   (b) Perform relational classification
   (c) Output a probability distribution
3. Apply ensemble classification to the resulting probability distributions of the relations
4. Apply ensemble classification to the output of the local classifier in 1. and the ensemble of relations in 3.
5. Output final classification

the value of an attribute for an instance is highly correlated with the same variable from a related instance [12]. *Collective classification* causes the propagation of information in a graph, so that instances with unknown labels, if they are initialized with a certain value, can be integrated in the classification process. Many studies [3, 21, 25] have shown, that *collective classification* may improve the classification accuracy significantly.

We use two types of *collective classification* procedures. For most *relational classification* methods, we apply *relaxation labeling*, like Macskassy and Provost [16, 17], originally introduced by Chakrabarti et al. [3]. In this algorithm each instance is initialized with a probability distribution (e.g. prior distribution or probability distribution returned by a local classifier). Then we iteratively classify each instance of the test set using a *relational classification* method M in the inner loop of the iterative algorithm. Within this iterative algorithm we use the classmembership probabilities of the neighborhood estimated in the previous iteration, whereas the output of the classification algorithm (a probability distribution) is used as input for the next iteration $t-1$:

$$P(c|x)^t = M(R_x^{(t-1)}) \tag{2}$$

The procedure stops when a certain number of iterations is reached or the algorithm converges. Here the uncertainty is kept and propagated in the graph.

The second iterative algorithm is a special case of *relaxation labeling* and is called *link-based classification* method, introduced by Lu and Getoor [15]. In their approach, the instances are initialized with a class label (e.g. prior class or the class assigned by a local classifier), so that the uncertainty is not considered in this algorithm. Like in *relaxation labeling* each instance is classified in each iteration and the output of the *relational classification* algorithm (a certain category) is used as input for the next iteration until convergence or a certain number of iterations is reached.

6.1. Simple Relational Methods

Now we present some simple relational methods. This methods are simple nearest neighbor procedures, which do not learn. The first method we have used, is the *probabilistic relational classifier (PRN)* introduced by Macskassy and Provost [16]. The classmembership probability of an instance $x \in X$ and a class $c \in C$ is computed as the weighted arithmetic mean of the classmembership probabilities of the neighbors x', whereas R_x is the set of neighbors of x:

$$P(c|x) = \frac{1}{Z} \cdot \sum_{x' \in R_x} w(x,x') \cdot P(c|x') \tag{3}$$

Whereas Z is a normalizing constant and $w(x,x')$ is the weight of the link between the instances x and x'.

Furthermore, we want to introduce two extensions of this algorithm, which we have developed. In our first extension, named *PRNGeometricMean*, we compute the classmembership probability of x to the class c by building the weighted geometric mean of the neighboring classmembership probabilities:

$$P(c|x) = \left(\prod_{x' \in R_x} P(c|x')^{w(x,x')} \right)^{1/\sum_{x' \in R_x} w(x,x')} \tag{4}$$

The weighted geometric mean is often a better way to represent probabilities, especially here, where we assume, that the classmembership probabilities of the neighbors given the instance x are independent of each other.

The second extension is our algorithm *PRN2Hop*, which additionally considers indirect neighbors at a distance of two hops, only if the following condition is fulfilled: The size of the neighborhood R_x of an instance x is less than a certain number d. Consequently additional neighbors are included in the classification process only for instances with a small neighborhood, this avoids noise. These approach addresses the sparsity problem of relations, which has been described in section 4. The classmembership probability is computed as follows:

$$\begin{aligned} P(c|x) \quad = \quad & \frac{1}{Z} \cdot \sum_{x' \in R_x} (w(x,x') \cdot P(c|x') + \\ & + \sum_{x'' \in R_{x'} || R_x | < d} w(x',x'') \cdot P(c|x'')) \end{aligned} \tag{5}$$

The optimal number of neighbors d for *PRN2Hop* has been determined on a small part of the training set.

Every method we described above is used together with *relaxation labeling*.

6.2. Relational Learning Methods

Furthermore we want to present some *relational classification* methods, which learn a model on the training instances. There exist several approaches to cope with relational data in order to learn a model. The most popular approach is, to propositionalize relational data by aggregating set-valued attributes. Propositionalization was often applied to relational data in order to "flatten" the data [15, 19, 21]. Kramer et al. [14] showed, that propositionalization may have no negative effect to classification accuracy.

In our research, we have analyzed several aggregation functions, which map the set-valued attributes (classes or probability distributions of the neighbors of a particular entity) to an aggregated value in order to use normal classification methods.

First, we have used *weighted average* as aggregation function, which calculates the attribute value by building the average of the weighted probabilities of the neighbors of an instance (which is equivalent to *PRN* in equation (3)). This is similar to the aggregation function COUNT [15], but using probabilities instead of counts and additionally normalizing the sum of weighted probabilities. After calculating the attributes for the training instances, we learn a model on these attributes with a learning method (similar to [15]).

The second aggregation function we present here, is the *Indirect RVS Score*, which was introduced by Bernstein et al. [1], not used as aggregation function for a learning algorithm, but as a similarity measure for a simple classifier. It is defined as follows:

$$sim(x,c) = \frac{\overrightarrow{x} \cdot \overrightarrow{ic}}{\|\overrightarrow{x}\| \cdot \left\|\overrightarrow{ic}\right\|} \tag{6}$$

The *Indirect RVS Score* is based on the relational vector space model [1] and corresponds to the Cosine Similarity measure. It computes the similarity among an instance vector $\overrightarrow{x}$, which contains the weights of the links from each instance to instance x, and the indirect class vector $\overrightarrow{ic}$, which is the sum over all instances that belong to the class c :

$$\overrightarrow{ic} = \sum_{x \in c} \overrightarrow{x} \tag{7}$$

The difference to the other approaches presented in this paper is, that the neighbors of an instance are not considered directly, but through the indirect class vector.

Moreover we have used a Naive Bayes Classifier based on the approach of Chakrabarti et al. [3] and Macskassy and Provost [17], however, instead of using *relaxation labeling*, we use *link-based classification* as *collective classification* algorithm. We call this algorithm *Weighted Naive Bayes*. In contrast to the classical Naive Bayes classifier, this approach considers the weights of the links between an instance and its neighbors. Here the attributes of an instance are the classes of its neighbors, while the order of these attributes is not important.

The classmembership probability for an instance x and class c is computed according to Bayes rule as follows:

$$P(c|x) = \frac{P(R_x|c) \cdot P(c)}{P(R_x)} \tag{8}$$

While $P(R_x|c)$ can be computed as:

$$P(R_x|c) = \frac{1}{Z} \cdot \prod_{x' \in R_x} P(c(x')|c)^{w(x,x')} \tag{9}$$

Z, R_x and $w(x,x')$ are defined as in Section 6.1. The advantage of Naive Bayes is its simplicity, which is the independence assumption. Although this can not hold for relational data, Domingos and Pazzani [6] showed, that Naive Bayes classifiers performed well. However, most Machine Learning algorithms cannot be applied to relational data, because each instance can have a different number of neighbors and many algorithms cannot cope with missing values. Naive Bayes however, support missing values, so that we can choose the number of attributes as the maximum existing number of neighbors. That means, we can consider the classes of the neighbors individually instead of aggregating them as in the former approaches.

Like for the simple relational methods, we use a *collective classification* algorithm for each presented relational learning method. First the train instances are initialized and the attributes have to be computed, then a model is learned. Next, the test instances are initialized, the attributes are computed and the learned model is applied in each iteration to all test instances until the algorithm converges or a certain number of iterations is reached. That means, in contrast to the EM (Expectation Maximization) algorithm, we learn the model only once and perform the iterations only on the test instances, on which the learned model is applied.

7. Ensemble Classification Methods

In this section, we present two *ensemble classification* methods we have used in our generic framework. We have decided to use *voting*, a simple *ensemble classification* technique and *stacking*, a more complex learning method. As mentioned in the introduction, *ensemble classification* may lead to significant improvement on classification accuracy. This is because uncorrelated errors made by the individual classifiers are removed by the combination of different classifiers [5]. *Ensemble classification* reduces variance and bias, moreover, the combination of several classifiers may learn a more expressive concept compared to a single classifier.

First, we sketch briefly simple unweighted *voting*: After performing the individual classifiers, we receive probability distributions for each classifier K_l as output and build the arithmetic mean of the class-membership probabilities for each test instance and class:

$$P(c|x) = \frac{1}{L} \cdot \sum_{l=1}^{L} P_l(c|x) \tag{10}$$

If the classmembership probability of one of the initial classifier is zero, we will use (10) on the remaining classifiers. The disadvantage of *voting* is, that *voting* performs well only if the results of the individual classifiers are similar.

The second *ensemble classification* method we present here is *stacking* [26], also named stacked generalization [28]. This method uses a meta-classifier to learn the probability distributions of the individual classifier and predicts

the probability distribution of the combination of these classifiers. Similarly to *voting*, we perform the individual classifiers K_l using k-fold cross validation first, these classifiers are called *level-0* classifiers. Then, we need to set up new instances, so called *level-1* instances, which contain the classmembership probabilities achieved by the individual classifiers and the original label c:

$$x_{new}=(P(c_1|x)_1,...,P(c_n|x)_1,...,P(c_1|x)_L,...,P(c_n|x)_L,c) \quad (11)$$

Then, a meta-classifier (we have used Logistic Regression) is learned on the new *level-1* training instances and the model is applied to the *level-1* test instances using k-fold cross validation.

As mentioned in section 5, we will use these methods in order to combine several relations, which as far as we know has never been done before with *ensemble classification*, and for one dataset we will additionally combine a local classifier with a *relational classifier*.

8. Evaluation and Experimental Results

This section deals with the question, whether *relational ensembles* are more accurate first, than complex relational models, which do not use *ensemble classification* and second, than models learned by a local classifier (text classifier). Moreover, we have compared our new approach of combining several relations with the approach of Macskassy and Provost [17]. We performed a systematic and extensive evaluation on two real life datasets. In total we have spent approximately 1500 CPU-hours for the experiments on these datasets. Some of them have been performed on a grid infrastructure.

Datasets To be comparable to other publications we applied our approach to *Cora* [18], a public dataset of scientific papers within the field of Computer Science. Similar to Macskassy and Provost [16], we have used the 4240 papers from the area of machine learning with 4012 unique authors. Additionally the dataset contains citations and each publication can be assigned to one of seven categories. In our experiments for *Cora* we have used two relations, *sameAuthor* and *citation*. For this dataset we have not considered content information.

The second dataset is the *CompuScience*[1] dataset, also from the domain of scientific papers within Computer Science. We have used 147571 papers with 117936 unique authors, 9914 reviewers and 2833 journals. For each publication, a title and an abstract is available, but citations are not. Thus, we will use the relations *sameAuthor*, *sameReviewer* and *sameJournal*. A characteristic of this dataset is, that each publication can be assigned to more than one category out of the 77 categories that are available. That means, we have to cope with the multilabel problem [24]. We used the One-vs-All approach, that means we have to solve $|C|$ binary classification tasks.

In table 1 some properties of the graphs for the different relations for *Cora* and *CompuScience* have been listed. There we can assess, that *CompuScience* is a much more complex dataset than *Cora*, especially the *sameJournal* relation is very complex, it contains millions of edges. Furthermore, we have observed, that the *relational autocorrelation* is less for the relations of *CompuScience* than for *Cora*. As mentioned in section 6, *relational autocorrelation* is an important property, which influences the accuracy of the methods we presented. *Relational autocorrelation* has been computed using CramersV Statistic [4].

Evaluation Metrics In this subsection, we describe shortly the evaluation metrics we have used. Similar to most researchers, who worked on the dataset *Cora*, where the multilabel problem does not exist, we have chosen accuracy as evaluation metric. For the second dataset, where we have to cope with the multilabel problem and solve binary classification tasks, we have decided to use Precision, Recall and F-Measure as evaluation metrics. First, these metrics are calculated per category and then aggregated by using the micro-average approach.

Experiments We have performed several experiments:

1. In the first experiment setting we have performed for both datasets three-fold cross validation using the algorithms we have presented in section 6, while the relations are combined by *ensemble classification*. The main question is: How will our algorithm perform compared to the complex methods RPT, RDN and RBC? Here we use accuracy as an evaluation measure. Despite of the existence of the multilabel problem for *CompuScience* dataset, we have considered only the main assigned class.

2. In the second experiment setup, we have performed holdout sampling on *Cora* (10 - 90 % splits) using the same algorithms as before. We wanted to know how our methods, which apply *ensemble classification*, perform in comparison to the results achieved by the approach of Macskassy and Provost [16] and how the amount of training data influences the accuracy.

3. For the *CompuScience* dataset we have performed an additional experiment, using three-fold cross validation. The main question is, how the ensemble of relational and local classifiers perform compared to the

[1] This corpora is extracted from a database produced by FIZ (Fachinformationszentrum) Karlsruhe for the information service io-port.net.

Table 1. Properties of graphs for Cora and CompuScience

	Cora		*CompuScience*		
	sameAuthor	*citation*	*sameAuthor*	*sameReviewer*	*sameJournal*
number of edges	28 134	11 258	1 032 069	750 113	54 938 662
avg. number of neighbors	13	5	13	10	744
number of singletons	636	657	23 224	97 255	441
relational autocorrelation	0.602	0.730	0.260	0.272	0.189

results achieved by a local classifier. In this experiment we consider all assigned classes.

Furthermore, we have done some experiments in order to compare *stacking* and *voting*, both, in the case of fusing relations and in the case of combining relational and local classifiers. These experiments showed, that *stacking* is almost always superior to *voting*. *Stacking* is used in the first and third experiment setting, in the second experiment setting this was not possible since *stacking* is performed with k-fold cross validation. Moreover, we could not apply *stacking* in the first experiment setting for the *CompuScience* dataset, because of high memory requirements. Regarding relational learning methods using aggregation functions, we performed several experiments with diverse classification models like Support Vector Machines, Decision Trees and Linear and Logistic Regression. Due to the good performance and the low computation time, we have decided to use Logistic Regression.

Concerning the combination of relations, for the *Cora* dataset, we have combined the relations *sameAuthor* and *citation*. The best combination for the *CompuScience* dataset is to consider all relations, but in the first experiment we have combined just the relations *sameAuthor* and *sameReviewer* since the complex methods RBC, RPT and RDN could not be applied to the *sameJournal* relation. Regarding *collective classification*, the instances in *Cora* dataset were initialized with the prior, because text has not been considered for this dataset. For *CompuScience* we initialized the test instances with the results achieved by a text classifier. We split the datasets into test and training sets by using stratified 3-fold cross validation or stratified hold-out sampling as mentioned before. The connections, among training and test set are preserved; this aids the inference process.

In the first two experiment settings we will compare *PRN* using the combination approach of Macskassy and Provost [16] to *PRN* using our approach of combining relations, this is called as from now *EPRN*, the other relational methods described in section 6, which use our relation combination approach are too prefixed with "E". The complex models RBC, RPT[2] and RDN[3] do not use *ensemble classification*. They represent each instance as a subgraph containing all its neighbors including several relations. The complex methods RBC, RPT and RDN have been performed using an open source tool called Proximity[4]. Because of the complex *CompuScience* dataset, we could not run the methods RPT and RDN on this dataset. All methods we have presented in section 6 and some others are included in our tool, named *RelEns*[5].

It uses Weka[6] , a machine learning workbench.

Experimental Results The results of the first experiment for *Cora* are depicted in figure 3. We have compared our algorithms to the complex methods RBC, RPT and RDN. We have observed, that four of our algorithms have achieved a significantly[7] higher accuracy (asterisked bars) than the mean accuracy. Furthermore we have noticed, that our extension of *EPRN*, *EPRN2Hop*, performed best.

Moreover, we have detected, that the simple methods, described in section 6.1 achieved higher accuracy than the *relational learning* methods. Only Logistic Regression using *weighted average* as aggregation function is also significantly better than the other listed methods. All of our methods performed better then the complex methods RBC, RPT, RDN. Our algorithms seem to profit from the strength of *ensemble classification*. Moreover they performed bet-

[2]The parameter values are set to tree depth=6,threshold=4, p=0.05.

[3]Applied with 300 iterations.

[4]Portions of this analysis were conducted using Proximity, an open-source software environment developed by Knowledge Discovery Laboratory at the University of Massachusetts Amherst (http://kdl.cs.umass.edu/proximity)

[5]Published on http://www.informatik.uni-freiburg.de/cgnm/software/index_en.html

[6]http://www.cs.waikato.ac.nz/ml/weka/

[7]Significance level is 0.05. The null hypothesis is, that the result of an algorithm is not higher than the other results. We used one sample t-test.

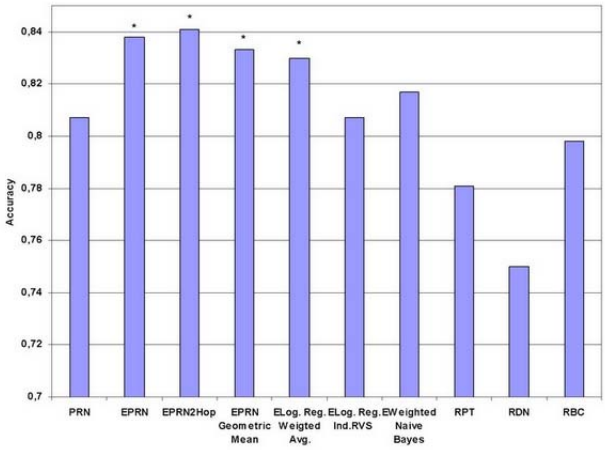

Figure 3. Accuracy of the combination of sameAuthor and citation (Cora)

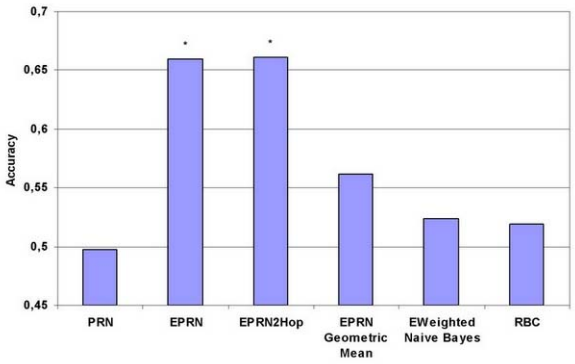

Figure 4. Accuracy of the ensemble sameAuthor and sameReviewer(CompuScience)

ter than *PRN* with the approach of Macskassy and Provost [16].

One interesting point is, to compare only the complex methods in order to capture the reasons for the low accuracy of RPT and RDN. Doing this comparison, one asses, that the simplest complex algorithm, RBC, achieves the best result. The poor results achieved by RPT could be due to the fact that it cannot exploit the power of *relational autocorrelation* because it does not use *collective classification*. Another reason could be, that we applied the algorithm with only one attribute, which is the category of the neighbors of an instance, but Neville and Jensen [22] showed, that 90% of the features constructed in their algorithm contain the category as an attribute. RDN, which performed better than RPT in [22], delivered poor results in our experiments although it uses *collective classification*. Regarding these results, one can argue, that the number of iterations has been chosen too low. We investigated this matter further by analyzing the influence of the numbers of iterations on the results. The experiments have shown, that the accuracy does hardly change after 50 iterations.

The fist experiment setting has been performed on the *CompuScience* dataset too. The results for *CompuScience* are depicted in figure 4. As mentioned before, we could apply only one complex method, namely RBC, on this dataset and we could combine just two relations, namely *sameAuthor* and *sameReviewer*. As for *Cora,* all methods listed in figure 4 have reached a higher accuracy than RBC and again the best result was achieved by *EPRN2Hop*. Furthermore, the accuracy of *EPRN* is much higher than the accuracy of *PRN* (relations combined by using the approach of Macskassy and Provost [16]). Because of the complexity of this dataset and consequently long computation time and high memory requirements, some of our methods could not be performed. So, the answer to the main question posed in the first experiment setting is that for both datasets the relational methods described in section 6 have performed better than the complex methods RBC, RPT and RDN.

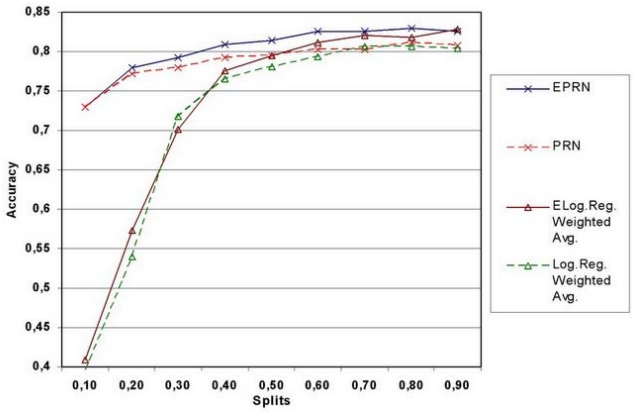

Figure 5. Accuracy holdout sampling (Cora)

The next experiment uses holdout sampling on *Cora* in order to compare the relation fusion by *ensemble classification* to the approach of Macskassy and Provost [16], who merged the relations and summed the weights of the relevant edges. Furthermore we want to analyze how the methods perform on different training and test set sizes. We have split the data in training and test sets of different percentage (10 - 90 %) and have performed the algorithms for each percentage on five randomly chosen splits. We applied both combination approaches to the *Cora* dataset. Figure 5 shows the accuracy achieved by *PRN* and Logistic Regression using *weighted average* as aggregation function on several splits. The solid line represents an algorithm, which

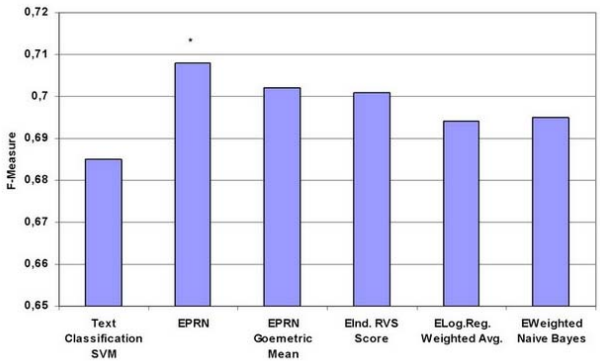

Figure 6. F-Measure of ensembles of text and relational classification compared to text classification

uses our fusion approach with *ensemble classification* (here *voting*), the dotted line shows the accuracy of the the algorithm, which uses the combination approach of Macskassy and Provost [16]. Both algorithms, *EPRN* and *ELogistic Regression* using the *weighted average* aggregation function have achieved significantly higher accuracy than they have done with the approach by Macskassy and Provost [16]. Moreover we observe, that the *relational learning* method achieves poor results when the amount of training data is small. While the performance of the simple relational method is much higher on few training examples.

As mentioned in the previous section, the last experiment setting has been applied on *CompuScience* dataset. Here, we have measured the performance of the algorithm with F-Measure. The aim of this experiment is to compare the ensembles of *relational classification* and text classification with pure text classification. We fused first, all relations using *stacking* and then combined the result with the outcome of a text classifier using *stacking* as well. As a local classifier, we have chosen a Support Vector Machine using the words of the abstract and title of a paper as features of the SVM. Figure 6 shows the F-Measure of some algorithms, which we have described in section 6. One can see, that the ensembles of *relational classification* and text classification always achieves higher F-Measure values than the text classifier (using only local features). The results of *EPRN* fused with the results of the text classifier are actually significantly better than the results of the other algorithms. The best *relational learning* algorithm in this experiment in contrast to the experiments on *Cora* is *ELogistic Regression* using *Indirect RVS Score* as aggregation function. This confirms the hypothesis of Bernstein et al. [1], that the *Indirect RVS Score* is able to achieve better results when the *relational autocorrelation* is lower.

As consequence of these experiments, we recommend to exploit the relations existing in the data, in order to improve accuracy.

9. Conclusions and Future Works

We have presented our generic *relational ensemble model*, that includes a new way of combining several relations using *ensemble classification* and considers relational features as well as local features using *ensemble classification*. In the "Evaluation and Experimental Results" section we have shown, that our approach improves classification accuracy significantly compared to complex *relational learning* algorithms [19, 20, 22] and to a local classifier. Furthermore, we have shown, that our way of combining relations is significantly better than the approach by Macskassy and Provost [16] and in most of the experiments our simple relational method *EPRN2Hop* performed best.

In future works we plan to do further experiments with new aggregation functions to improve the performance of *relational learning* algorithms and evaluate other *ensemble classification* methods. Furthermore we will compare our approach to a classifier considering local and relational features as well.

10. Acknowledgements

We thank Ute Rusnak of Fachinformationszentrum (FIZ) Karlsruhe for providing the dataset *CompuScience*. Furthermore we thank Robert Koppa for the implementation of the text classifier we used.

This work was funded by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978.

References

[1] A. Bernstein, S. Clearwater, F. Provost, The Relational Vector-space Model and Industry Classification, In Proceedings of IJCAI Workshop on Statistical Models from Relational Data, 2003, pp. 8-18.

[2] Breiman, L.: Bagging Predictors. In Machine Learning 24, 1996, pp. 123-140.

[3] S. Chakrabarti, B. Dom, P. Indyk, Enhanced Hypertext Categorization Using Hyperlinks. In Proceedings of ACM SIGMOD,1998, pp. 307-318.

[4] Cramer, H.: Mathematical Methods of Statistics. Princeton University Press, 1999.

[5] T.G. Dietterich, Machine Learning Research: Four Current Directions, AI Magazine 18(4), 1997, pp. 97-136.

[6] P. Domingos, M. Pazzani, On the Optimality of the Simple Bayesian Classifier under Zero-one Loss. Machine Learning 29, 1997, pp. 103 - 130.

[7] J. Fürnkranz, Hyperlink Ensembles: A Case Study in Hypertext Classification, In Information Fusion 3, 2002.

[8] A. Heß, N. Kushmerick, Iterative Classification for Relational Data: A Case Study of Semantic Web Services, In Proceedings of European Conference on Machine Learning, 2004.

[9] J. Hopfield, Neural Networks and Physical systems with Emergent Collective Computational Abilities, National Academy of Science 79, 1982, pp. 2554-2558.

[10] Z. Huang, H. Chen, D. Zeng, Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering, In ACM Transactions on Information Systems 22(1), 2004, pp. 116-142.

[11] D.Jensen, J. Neville, B. Gallagher, Why Collective Inference Improves Relational Classification, In Proceedings of the 10th ACM SIGKDD, 2004.

[12] D. Jensen, J. Neville, Linkage and Autocorrelation Cause Feature Selection Bias in Relational Learning, In Proceedings of the 19th International Conference on Machine Learning, 2002.

[13] J. Kleinberg, Authoritative Sources in Hyperlinked Environment, In Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, 1998.

[14] S. Kramer, N. Lavrac, P. Flach, Propositionalization Approaches to Relational Data Mining, In Relational Data Mining, Kluwer, 2001, pp. 262-291.

[15] Q. Lu, L. Getoor, Link-based Text Classification, In Proceedings of IJCAI Workshop on Text Mining and Link Analysis, 2003.

[16] A.S. Macskassy, F. Provost, A Simple Relational Classifier, In Proceedings of the Multi-relational Data Mining Workshop ACM SIGKDD, 2003.

[17] A.S. Macskassy, F. Provost, Classification in Networked Data: A Toolkit and a Univariate Case Study, CeDER Working Paper, Stern School of Business, New York University, 2004.

[18] A. McCallum, K. Nigam, J. Rennie, K. Seymore, Automating the Construction of Internet Portals with Machine Learning, In Proceedings of Information Retrieval 3(2), 2000, pp. 127-163.

[19] J. Neville, D. Jensen, B. Gallagher, R. Fairgrieve, Simple Estimators for Relational Bayesian Classifiers, In Proceedings of the 3rd IEEE International Conference on Data Mining, 2003, pp. 609-612.

[20] J. Neville, D. Jensen, L. Friedland, M. Hay, Learning Relational Probability Trees, In Proceedings of SIGKDD, 2003.

[21] J. Neville, D. Jensen, Iterative Classification in Relational Data, In Proceedings of AAAI Workshop on Learning Statistical Models from Relational Data, 2000, pp. 13-20.

[22] J. Neville, D. Jensen, Collective Classification with Relational Dependency Networks, In Proceedings of the second Workshop on Multi-relational Data Mining KDD, 2003.

[23] F. Provost, P. Domingos, Well-trained PETs: Improving Probability Estimation Trees. CeDER Working Paper, Stern School of Business, New York University, 2000.

[24] R.M. Rifkin, A. Klautau, In Defense of One-Vs-All Classification Journal of Machine Learning Research 5, 2004, pp. 101-141.

[25] B. Taskar, E. Segal, Koller, D., Probabilistic Classification and Clustering in Relational Data, In Proceedings of the 17. International Joint Conference on Artificial Intelligence, 1997, pp. 870-878.

[26] K.M. Ting, I.H. Witten, Stacked Generalization: When does it work?, In Proceedings of the International Joint Conference on Artificial Intelligence, 1997, pp. 866-871.

[27] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Publishers, 2000.

[28] D.H. Wolpert, Stacked Generalization, In Neural Networks 5, Pergamon Press, 1992, pp. 214-259.