Automatic Classification for the Identification of Relationships in a Meta-Data Repository

Gerd Beuster   Ulrich Furbach   Margret Gross-Hardt   Bernd Thomas

{gb,uli,margret,bthomas}@uni-koblenz.de
Universität Koblenz-Landau, Institut für Informatik

Abstract. For a large company a prototype for automatic detection of similar objects in database systems has been developed. This task has been accomplished by transferring the database object classification problem into a text classification problem and applying standard classification algorithms. Although the data provided for the task did not look promising due to the small number of positive examples, the results turned out to be very good.

1 Introduction

Large companies manage huge amounts of data (i.e. data about their customer base, their suppliers, products etc.). Usually, there are many databases and applications that store and provide these data. Since these databases have been developed and managed independently, they are often heterogeneous in logical structure, attribute naming and semantics. Nowadays, companies face the need for an integrated view on their data. That is, they want to understand the relationships between data in different databases or between applications using the same database. Detecting similar objects and relationships between objects is a crucial integration task in enterprise application integration and business-to-business applications. In order to achieve these goals, heterogeneities within the data stored in different databases have to be recognized or even resolved. There are various approaches how to deal with heterogeneous data sources. Some are more tightly coupled and define common views on multiple data sources, whereas more loosely coupled approaches maintain the autonomy of distributed databases [10, 2].

A recent development is the creation of *meta-data repositories* to manage meta-data about systems, databases and the data therein [4]. Meta-data repositories play the role of information brokers and provide applications and users with the information necessary to determine dependencies between data sources. A meta-data repository contains information about objects, tables and relationships in the various databases used in a company. It uses this information to analyze business processes, to provide information about marketing campaigns etc. The amount and accuracy of the meta data is critical for the quality of the meta-data repository. Since most databases used in a company are developed independent of each other, information about the same real world entity—e.g customer name and address information—is stored multiple times at different places, under different names and in different formats. In order to make the conglomerate of different database structures manageable and to avoid inconsistencies, these kind of dependencies should be detected and stored in the meta-data repository.

The success of a meta-data repository is based on the accuracy and completeness of the data. It has to be maintained continuously. This requires a lot of additional work, because meta-data usually is maintained manually. When a set of new objects is to be added to the meta-data repository, an expert uses her knowledge about the meta-data repository and the new objects in order to add it.

2 Motivation

In order to reduce the amount of work needed to maintain a meta-data repository, the company wizAI did a study on how this process can be automated.

To illustrate the basic problem and our approach consider Figure 1. It is important to point out, that not the table rows or attribute names used within an application are stored as attribute values of an meta-data object instance. But the administrative repository user determines the attribute values of the meta-data object describing an application or certain data of interest stored in this application. This is also the reason why the number of relationship types (*reference types*) and the actual attribute values are not fixed, because a meta-data repository is in almost all cases under constant change. In our case the company is running more than 150 different applications where each application's data-scheme can change from time to time. Though we can assume that the meta-data classes / objects are fixed according to their attributes, we do not know very much about some of the possible attribute values, except their data type. As for example attributes like *description* can take upto 2000 characters of free formulated text. Whereas attribute values for the attribute *object_type_id* are only known regarding the already inserted instances of the class *object_type_id* (not shown in the figure). For examples, the *object_type_id: 1* stands for "*application + database*". Although this might yield important additional information, in a first step, we focused only on the information given by the attribute values of the meta-data objects. So, we turned our attention only on two data sources of the given meta-data repository, the *object* instances and the *reference* instances, to develope an approach for the automatic identification of relationships for objects that have to be added to the repository. Besides the two mentioned classes there are other additional instances of classes like *reference type id*, *object type id*, *attribute type* etc. stored in the meta-data repository.

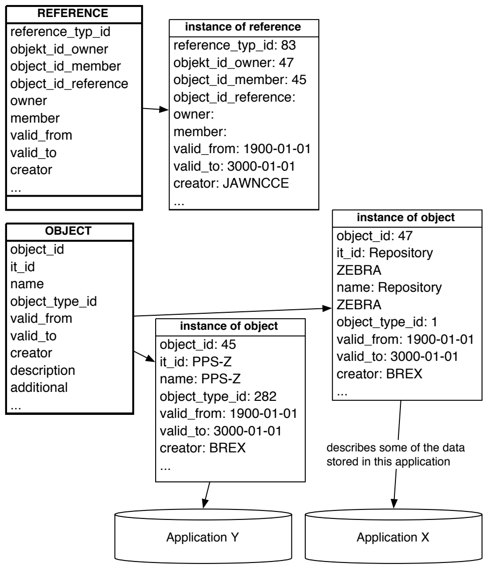

Fig. 1. sketch of a meta-data repository

The basic system idea is to learn automatically classifiers, which are able to determine if certain meta-data objects are in relationship to each other, and if so of which type these relationships are. From a conceptual point of view this appeared to be a standard classification problem as known in the machine learning area, but with a closer look on the none finite, none disjunctive attribute values, we were encouraged to start first experiments with our readily available text classification system MIC [1]. It turned out that after some transformations of the input data, the standard text classification methods were able to identify dependencies between meta-data objects with very promising results.

Thus we developed a prototypical system within the study, that aids the user in adding objects plus the right relationships with existing objects already stored to the meta-data repository. Whenever a new object has to be added to the repository, the automatic classification system lists potential relationships of the object to other objects. Among these proposals the user has to decide if the suggestions of the system are correct. This user task requires a lot less expert knowledge and time than a complete manual maintenance.

3 Problem Definition

The classification task was to identify relationships between meta-data objects. We were provided with a list of relationship types, with descriptions of meta-data objects, and with some information about relationships between objects as shown in Figure 1.

We will use the following terms: *Object* for an entity of a domain, e.g. a system, data model, business object, etc. Let $O_1, \dots, O_n$ be objects. Each object has a number of characteristics $A_1, \dots, A_m$ called *attributes*. An object is characterized by its attributes. We write this characterization as $< O_i.A_1, \dots, O_i.A_m >$. Objects of the same kind are of the same *object type*. The type of an object is an attribute of the object. In the following, we deal with *typed relationships* R over a set of objects $S = \{O_1, \dots, O_n\}$. R is a subset of $\{(O_i \times O_j \times t) | O_i, O_j \in S, 1 \leq t \leq n\}$. We call the third element of the triple the *type* of the relationship. n is the total number of relationship types. We are only dealing with typed relationships in the rest of this paper.

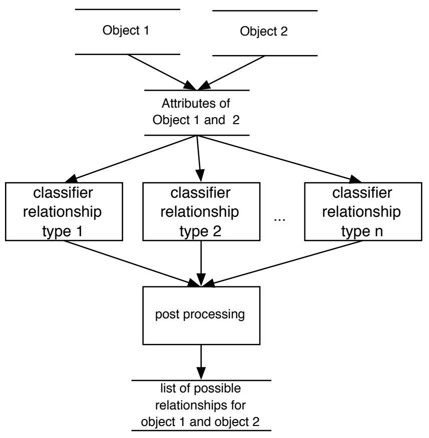
Object 1
Object 2
Attributes of Object 1 and 2
classifier relationship type 1
classifier relationship type 2
...
classifier relationship type n
post processing
list of possible relationships for object 1 and object 2

Fig. 2. Classification system scenario 1

It should be noted that the database objects are descriptions of data types, applications or business objects etc. and not actual instances of database entries. A *meta-data repository* is a set of objects, together with a set of typed relationships over these objects: $M = (S, R)$.

To support the user in identifying relationships between database objects, two typical usage scenarios were identified.

The first scenario models the situation where the user has identified two objects which might be in one or more relationships to each other. The system should make suggestions about the potential kinds of relationships between the objects. Since the number of relationship types is fairly large (in the example data provided by the client, there were about 100 relationship types), an aid of this kind can significantly speed up the process of updating the meta-data repository with new data. This can be formally defined as follows:

Scenario 1 *Given two objects O_1 and O_2, and a meta-data-repository $M = (S,R)$, find all potential relationships P that might hold between O_1 and O_2, and present them to the user. The user selects $P' \subseteq P$. An extended repository M' is created by adding these relationships: $M' = (S \cup O_1 \cup O_2, R \cup P')$.*

In the following we assume independence between the relationship types. This means, in order to decide if a given object is in a certain relationship with some other object, we do not take into consideration if the object is in some other relationship with the same or a third object. Therefore, we can treat each relationship type separately. This allows us to reduce the problem of scenario 1 as follows: Given two objects, decide if they are in the given relationship or not. Thus, the input for the classifier is a tuple of two objects (O_1, O_2), and the output a binary value indicating whether the two objects are in relationship with each other or not. It would also be possible to give a confidence value indicating how likely it is that the two objects are in the given relationship.

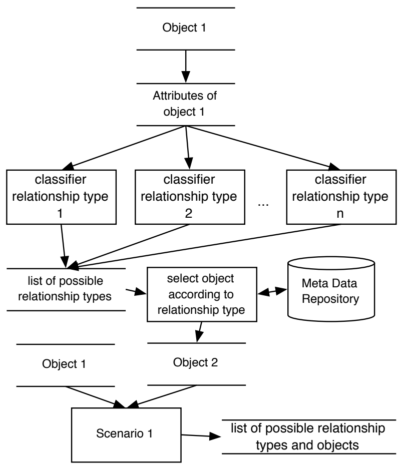
Object 1
Attributes of object 1
classifier relationship type 1
classifier relationship type 2
...
classifier relationship type n
list of possible relationship types
select object according to relationship type
Meta Data Repository
Object 1
Object 2
Scenario 1
list of possible relationship types and objects

Fig. 3. Classification system scenario 2

For scenario 1 (see Figure 2) the objects are combined and transfered into a feature representation. This feature vector is then classified by each of the classifiers for the various relationship types. Note that these classifications can be done in parallel for all relationship types. The classification results of all classifiers are then presented to the user.

In scenario 2, the user presents a single object to the system, and the system returns a list of other objects from the meta-data repository together with the potential relationships between the given objects and the objects found by the system. Formally this is defined as follows:

Scenario 2 *Given an object O_1 and a meta-data-repository $M = (S,R)$, find all potential relationships P that might hold between O_1 and some $O_2 \in S$, and present them to the user. The user selects $P' \subseteq P$. An extended repository M' is created by adding these relationships: $M' = (S \cup O_1, R \cup P')$.*

There is an obvious relationship between scenario 2 and scenario 1: In scenario 2, each object from the meta-data repository is combined with the new object and run through scenario 1. But following this naïve approach is not advisable for two reasons: Since the new object has to be checked against all existing objects for all relationship types, with N objects and R relationship types in the meta-data repository, $N \times R$ classification would be necessary for each new object. This would unnecessarily increase the complexity of the classification process. For this reason, the classification process in scenario 2 is split into two parts (depicted in Figure 3): In the first step, the object is used as input data of a relationship type classification process. In the second step, the classification system draws each of the objects from the meta-data-repository (in Figure 3 shown as object 2), and feeds the combination of the two objects into all the classifiers for the relationship types from step one (scenario 1).

Since the new object has to be checked against all other objects, the number of classifications to compute is proportional to the number of objects already in the meta-data repository. So the difference is that the new object is not checked for all relationship types with all other objects, but also for a reduced set of relationship types which look promising. Thus, for scenario 2 we learn n binary classifiers, one for each relationship type to use the provided classification information in step 2. Which is than used for the selection of potential relationship partners regarding the given object.

4 Transforming the problem into a text classification problem

Section 2 already gave a sketch of our approach for identifying relationships in a meta-data repository. But so far the only argument to use a text classification system for this task was its availability. Having a closer look at the data of the repository and the properties of the object attribute values leads to the assumption, that a text classification system in general seems to be a good choice. Since standard classification methods take vector representations of the objects to be classified as inputs, it would be reasonable to use the vector representation of the meta-data objects (Section 3) directly as the input data for classification algorithms. But this representation is not well suited for our classification task, because the nature of the attribute values (none disjunctive and none finite sets of possible values, as illustrated in Section 2) makes is hard or nearly impossible to find a binary encoding following the standard transformation using a binary attribute for each possible value of each attribute.

Therefore the main idea of our approach is to assume that database objects are similar to text objects and thus can be handled by our system for automatic text classification (MIC [1]). We used a well known method from text classification: the attribute values of objects are treated as texts. For the classification tasks in scenario one and the second step in scenario two, the input data for the classifiers are combinations of two objects. In these cases, the textual representations of the two objects are concatenated. Whereby the textual representation of an object is created by treating attribute values as strings and concatenating all attributes of the object, where each attribute string is separated by a whitespace-character.

As with standard classification, text classification also requires a text to be transformed into a vector representation. We use the relative frequency of words in a text as the input feature vector. For each word in the vocabulary of the training data set, the relative frequency of the word in the textual representation of the object is calculated.

This is the number of appearances of the word in the text, divided by the total number of words in the text. This is a common method for text representation, described e.g. in [6, page 183]. Since we need fixed vector sizes for some of the classifiers, only the $n = 100$ most informative words for the classification task, determined according to Shannon's formula [9], are used in the vector representation of the text.

5 Learning and Results

For this project, we used three standard classification methods: Naïve Bayes Classifiers [5], ID-3 Decision Trees [7] and fully connected feed-forward, one hidden layer, backpropagation Neural Networks [8]. We assume the reader to be familiar with these machine learning methods.

Two data sets were provided for this classification task. The first data set contained 3579 objects, 79 different relationship types, and 18605 relationships between objects. The second, considerably smaller data set, contained 748 objects, 92 different relationship types, and 1080 relationships between objects. Since we used separate classifiers for each of the relationship types, we had on average 235 instances of relationships between two objects for the first data set, and 12 instances of relationships between two objects for the second data set.

scenario 1							
test	e	tp	precision	recall	F1	fp	fn
S1,N2	8997	557	98,41	97,55	97,98	9	14
S2,N2	18287	549	79,57	78,54	79,05	124	150
S3,N2	18090	525	75,54	83,20	79,19	170	106
scenario 2							
S1,N1	2193	660	94,82	95,65	95,23	36	30
S1,N2	9024	698	96,67	84,61	90,24	24	127
S2,N1	2254	671	93,32	92,94	93,13	48	41
S2,N2	10974	718	97,28	81,13	88,47	20	167
S3,N1	2218	645	93,89	96,56	95,21	42	23
S3,N2	11026	672	97,39	85,50	91,06	18	134

Fig. 4. Decision Tree results

Since there were on average 235 relationship instances (data set one) resp. 12 relationship instances (data set two) for each relationship type, we had $\approx 6.57\%$ resp. $\approx 1.60\%$ positive examples in the data sets. This got even worse for scenario 1, because each object can be combined with every other object. For the first data set, there are 6402831 combined objects. Thereby, the number of positive examples was reduced drastically to $\approx 0.004\%$. For the second scenario, we had 279378 combined objects resulting in a ratio of $\approx 0.004\%$ positive examples. This drastic disproportion between positive and negative examples did not allow us to use automatic classification algorithms successfully on these data sets. Therefore we used two additional methods to alleviate the disproportions in the data sets:

Only objects of the same type: A constraint of the meta-data repository is that only objects of the same type can be in a given relation. There are 67 object types in the first data set, and 74 object types in the second data set.

Restricted number of negative examples: The number of negative examples used for the training of classifiers were limited to 500 (S1), 2000 (S2), and 5000 (S3) examples.

For training and testing the positive data was split randomly into two halves and negative examples were added to the training and testing data sets until the desired size

(S1 to S3) of the data sets was reached. Negative examples were selected according to the following classes for scenario 1 and 2:

class	*scenario 1*	*scenario 2*
N3	The the set of all object pairs, minus the positive examples.	The set of all objects not in the set of positive examples.
N2	Object pairs who are in a relation, but not in the relation the classifier is trained for. N2 is a subset of N3.	Objects who are in relation, but not in the relation the classifier is trai:ned on.
N1	Only object pairs of the same object types as the objects in the positive examples. N1 is a subset of N2.	Objects of the same type as the objects in the positive examples.

For each scenario, type of object selection, and example size, the quality of the classifier was calculated. Since we assume independence between relationship types, these calculations were done independently for each relationship type (see Section 3). Results were accumulated over all relationship types. Figure 4 shows the accumulated classification results of the learned decision tree classifiers. Where *e* is the total number of examples in the testing data; *tp* the number of positive examples in *e*; *precision* the percentage of correctly classified positive examples in all examples that were classified positive, *recall* the percentage of correctly classified positive examples in all positive examples; *F1* [3] (harmonic mean of precision and recall); *fp* the number of negative examples which have been classified wrongly as positive; *fn* number of positive examples which have been classified wrongly as negative.

Some test constellations were not applicable, because for some class of examples the data set is empty. In some cases for any relationship type, there are no objects O_1 and O_2 such that O_1 and O_2 are of the same object type as the objects in the set of positive examples, and in some relationship to each other, but not in the relationship that should be learned (e.g. test scenario 1 and setting [S1,N1]). Figure 5 shows the average of the F1 values for each classification method and the two scenarios.

	d.t.	n.b.	n.n.
scenario 1	85,40	70,25	84,02
scenario 2	92,22	90,69	92,13

Fig. 5. Average of F1 results of all tests

6 Conclusion

Existing machine learning techniques for text classification proved to be surprisingly robust for the application to a very different task. The provided data for this special classification task was not well suited for three reasons: It was not actual text, but descriptions of meta-data objects. There was a blatant disproportion between positive and negative examples. Using *S1* for scenario 1 and arbitrary settings for scenario 2 all precision and recall values, have been between 71% and 98,5%, which are promising results due to the fact of using standard learning algorithms for this task.

It should be noted that a good precision value alone does not prioritize one learning technique over the other. In the scenarios we illustrated in this paper, potential relationships are presented to a human user who decides whether the system's suggestions is correct (it is an relationship indeed) or not (there is no relationship). In these scenarios, false negative (existing relationships that are not detected) are a lot worse than false positives (non-existing relationships that are detected wrongly). False negatives literally get lost: They are not presented to the user, and therefore can not be added manually. On the other hand, false positives are presented to the user, so she can remove them.

Based on these considerations, we can conclude that the optimization by restricting the negative example sets according to N1 yields the overall best results (for all three techniques increased recall values from 3% to 11%). Obviously this also results in decreased precision values upto 10%, which is acceptable due to the improved recognition of positive examples. Comparing all three techniques according to their F1 values (Figure 5) the decision tree classifiers showed the best classification results.

Using the presented classification system reduced the necessity for human intervention drastically. Still, the system is not completely autonomous. Although the error rate is fairly low, it is still advisable and necessary to let a human review the suggestions of the system. Beside reducing the amount of human intervention, the human supervisor also needs less expert knowledge than a human classifying database objects unaided. When using a decision tree algorithm for classification, the researcher additionally can get insights into the structure of the data, and may develop meta-knowledge about what kinds of objects are in relationships. We think the approach presented in this paper can be improved further by changing the representation of the objects. So far, we treat them as plain text, ignoring all structural information (e.g. additional information about relationship types, object ids that are present in the meta-data repository, other relationships an object has). We expect improved behavior from better representations which preserve the structural information of the objects.

References

1. G. Beuster. MIC — A System for Classification of Structured and Unstructured Texts. Master's thesis, University Koblenz, 2001. http://www/˜gb/papers/thesis_mic/mic.pdf.
2. A. Bouguettaya, B. Benatallah, and A. K. Elmagarmid. *Interconnecting Heterogeneous Information Systems*. Kluwer Academic Publishers, 1998.
3. N. Chinchor. Muc-4 evaluation metrics. In *Fourth Message Understanding Conference*, pages 22–29. Morgan Kaufmann, 1992.
4. D. Marco. *Building and Managing the Meta Data Repository: A Full Lifecycle Guide*. John Wiley & Sons, 2000.
5. M. Maron. Automatic indexing: An experimental inquiry. *Journal of the ACM (JACM)*, 8:404–417, 1961.
6. T. M. Mitchell. *Machine Learning*. McGraw-Hill International Editions, 1997.
7. J. Quinlan. Discovering rules by induction from a large collection of examples. In D. Michie, editor, *Expert systems in the Micro-Electronic Age*, pages 168–201. Edinburgh University Press, Edinburgh, 1979.
8. D. D. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, pages 533–536, 1986.
9. C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.
10. A. Sheth and J.Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22:183–236, 1990.