Choosing the Right Patterns

An Experimental Comparison between Different Tree Inclusion Relations

Jeroen De Knijf [*] and Ad Feelders

Algorithmic Data Analysis Group
Department of Information and Computing Sciences, Universiteit Utrecht
PO Box 80.089, 3508 TB Utrecht

Abstract. In recent years a variety of mining algorithms has been developed, to derive all frequent subtrees from a database of labeled ordered rooted trees. These algorithms share properties such as enumeration strategies and pruning techniques. They differ however in the tree inclusion relation used and how attribute values are dealt with. In this work we investigate the different approaches with respect to 'usefulness' of the derived patterns, in particular, the performance of classifiers that use the derived patterns as features. In order to find a good trade-off between expressiveness and runtime performance of the different approaches, we also take the complexity of the different classifiers into account, as well as the run time and memory usage of the different approaches. The experiments are performed on two real datasets. The results show that significant improvement in both predictive performance and computational efficiency can be gained by choosing the right tree mining approach.

1 Introduction

Frequent tree mining has become an important and popular problem in the field of knowledge discovery and data mining. The main reasons for the increase in interest are the growing amount of semi-structured data (e.g. XML databases) and the urge to analyze and mine these databases. Furthermore, the availability of tree mining algorithms to exploit these databases, without losing information on the structure of the data, has increased the interest of the research community. Frequent tree mining can be seen as an extension of the apriori algorithm [1], to handle tree structured data in the mining process. Briefly, given a set of tree data, the problem is to find all subtrees that satisfy the minimum support constraint, that is, all subtrees that occur in at least $n\%$ of the data records.

Due to the popularity of tree mining, different approaches to mine labeled ordered trees have been proposed, see for example [2, 9, 12, 13]. Although, these methods share enumeration strategy and pruning techniques, the fundamental differences lie in the different notions of when a tree matches another tree and how attributes are handled. Also, when the notion of rooted ordered trees is broadened to unordered rooted trees, the

[*] Supported by the Netherlands Organisation for Scientific Research (NWO) under grant no. 612.066.304.

existing mining algorithms [3, 10, 14] use different tree inclusion relations. The fundamental question arises which of the mining algorithms should be used, when analyzing tree structured data. Comparisons between these methods are mainly based on time and memory performance. But besides run time and memory usage, the effect of the different tree inclusion relations and different approaches to handling attributes are of great importance to the data analysis task. To compare the different approaches, we consider the task of building a classifier from the frequent patterns produced by the tree miner. We consider several aspects in the comparison: accuracy and complexity of the classier produced, as well as runtime and memory usage of the tree miner.

In the next section we describe the basic notations and formally define the different tree inclusion relations and the different ways to handle attribute value-pairs in the mining process. In the following section the research questions are formulated. In section 4 we describe how a classification model is constructed from frequent patterns. In section 5 the experiments to answer the questions are carried out and evaluated. In the final section, we answer the stated questions and discuss the conclusions drawn.

2 Preliminaries

In this section we provide the basic concepts and notation used in this paper. A labeled rooted ordered tree $T = \{V, E, \leq, L, v_0, M\}$ is an acyclic directed connected graph which contains a set of nodes V, and an edge set E. The labeling function L is defined as $L : V \rightarrow \Sigma$, i.e., L assigns labels from alphabet Σ to nodes in V. The special node v_0 is called the root of the tree. If $(u, v) \in E$ then u is the parent of v and v is a child of u. For a node v, any node u on the path from the root node to v is called an ancestor of v. If u is an ancestor of v then v is called a descendant of u. Furthermore there is a binary relation '$\leq$' $\subset V^2$ that represents an ordering among siblings.

In some tree structured data, such as XML, attributes are used to describe properties of nodes. To model this, we assume a set of attribute-value pairs, denoted by : $\mathcal{A} = \{(A_1 : a_1), \ldots, (A_n : a_n)\}$, where each attribute-value has a finite domain. We further assume that there is an ordering specified on the attribute-value pairs, which can be arbitrary. To each node v in V, a subset of $\mathcal{A}$ is assigned; we call this set the attributes of v. More formally, we define a mapping $M : V \rightarrow \mathcal{P}(\mathcal{A})$. The size of a tree is defined as the number of nodes it contains; we refer to a tree of size k as a k-tree.

The notion of tree inclusion is dependent on how to handle attributes, up till now three different approaches are used:

- No attributes are associated with the data, or the attributes are considered irrelevant and are ignored in the mining process [2].
- No distinction is made between attributes and nodes; attribute-value pairs are added as child nodes of the corresponding original node [13].
- Attributes are modeled as properties of nodes. If a node has attributes associated to it in the database, then this node can only occur in a frequent pattern if the combination of the node with at least one of its attribute-value pairs is frequent [9].

In the remainder of this work, we will refer to these three cases as NOATR, NAIVE and ATR respectively. The two most commonly used tree inclusion notions, that handle attributes as described in the first two cases are:

Definition 1 *Given two labeled rooted trees T_1 and T_2 we call T_2 an* induced *subtree of T_1 and T_1 an* induced *supertree of T_2, denoted by $T_2 \preceq_i T_1$, if there exists an injective matching function Φ of V_{T_2} into V_{T_1} satisfying the following conditions for any $v, v_1, v_2 \in V_{T_2}$:*

1. *Φ preserves the labels: $L_{T_2}(v) = L_{T_1}(\Phi(v))$.*
2. *Φ preserves the order among the siblings: if $v_1 \leq_{T_2} v_2$ then $\Phi(v_1) \leq_{T_1} \Phi(v_2)$.*
3. *Φ preserves the parent-child relation: $(v_1, v_2) \in E_{T_2}$ iff $(\Phi(v_1), \Phi(v_2)) \in E_{T_1}$.*

Definition 2 *Given two labeled rooted trees T_1 and T_2 we call T_2 an* embedded *subtree of T_1 and T_1 an* embedded *supertree of T_2, denoted by $T_2 \preceq_e T_1$, if there exists an injective matching function Φ of V_{T_2} into V_{T_1}, satisfying the conditions 1 and 2 of definition 1. Additionally, Φ has the following property for any $v_1, v_2 \in V_{T_2}$:*

3'. Φ preserves the ancestor-descendant relation: if $(v_1, v_2) \in E_{T_2}$ then $\Phi(v_1)$ is an ancestor of $\Phi(v_2)$ in T_1.

The induced tree inclusion notion is used in the tree mining algorithm FREQT by Asai et. al [2] and the work by Termier et al. [12]. The embedded subtree relation is used in work by Zaki [13].

In the case were attributes are modeled as properties of nodes, an additional criterion to the definitions 1 and 2 is that the subtree relation should preserve the attributes; i.e., we add to definitions 1 and 2:

4. $\forall v \in V_{T_2}$: if $M(v) \neq \varnothing$ then $M(v) \subseteq M(\Phi(v))$.

These subtree relations are used in previous work of one of the authors [9].

In the remainder of this paper we use $\preceq$ to denote either an induced or embedded subtree relation. Let $D = \{d_1, \ldots, d_m\}$ denote a database where each record $d_i \in D$, is a labeled rooted ordered tree. For a given labeled rooted ordered tree T we say T occurs in a transaction d_i if T is a subtree of d_i. Let $\sigma_{d_i}(T) = 1$ if $T \preceq d_i$ and 0 otherwise. The support of a tree T in the database D is then defined as $\psi(T, D) = \sum_{d \in D} \sigma_d(T)$, that is the number of records in which T occurs one or more times. T is called frequent if $\psi(T, D)/|D|$ is greater than or equal to a user defined minimum support ($minsup$) value. The goal of frequent tree mining is to find all frequently occurring subtrees in a given database. Notice that ψ is an anti-monotone function: $T_i \preceq T_j \Rightarrow \psi(T_i, D) \geq \psi(T_j, D)$. The anti-monotonicity property of ψ is used to efficiently compute all the frequent subtrees of a database.

3 Research Question

The first question we consider is whether the predictive performance of a classifier constructed from the patterns is better when attributes are included in the mining process. In previous work we claimed that the NAIVE approach produces lots of uninformative patterns [9]. In particular, if a node has attributes associated to it in the database, a frequent pattern in which this node occurs without attributes is uninteresting. Clearly, following the NAIVE approach, for every frequent pattern where some nodes have attributes there is a corresponding frequent pattern without attributes attached to the node.

Furthermore, every pattern from either the NOAT or the ATR approach is also a pattern of the NAIVE approach. As such, one can expect that the predictive performance achieved by the NAIVE approach, is as least as good as the predictive performance of the other two approaches. So, the question arises if the patterns excluded by the "at least one attribute per node" constraint are valuable in terms of predictive performance. Finally, what is the added value of the embedded tree inclusion relation? Every pattern derived with the induced tree inclusion relation is also part of the patterns derived with the embedded tree inclusion relation. As such, in the ideal case the difference in performance between the induced and the embedded approach is caused by the patterns that are embedded patterns only. Summarizing, the questions are what effect it has on the different performance measures whether we:

1. Include attribute-value pairs or not.
2. When including attribute-value pairs; use the NAIVE or the ATR approach.
3. Use the induced or the embedded tree inclusion relation.

4 Methodology

To compare the different approaches on the selected performance measures, we first need to construct the classifiers. In this section we describe the procedure that, given the tree mining algorithm, constructs a classification model.

In general, the goal of frequent tree mining algorithms is to find all frequently occurring subtrees in a database. In case of constructing a classifier the objective is to find discriminative patterns, i.e. patterns that discriminate between the different classes. To do so, we use the support-confidence framework. Given a dataset $D = \{d_1, \ldots, d_m\}$ consisting of m records, each record belongs to exactly one class,where the class label is assigned from the set of class labels $C = \{c_1, \ldots, c_k\}$. With D_{c_i} we denote the set of records in the database that has class label c_i, likewise with $D_{\overline{c_i}}$ the set of records in the database is denoted that has a class label different from c_i. The first step of finding discriminating patterns, is to compute all frequently occurring subtrees within a class. A tree T is called frequent within the class c_i if $\psi(T, D_{c_i})/|D_{c_i}|$ is greater than or equal to a user defined minimum support ($minsup$) value. The computation of all frequent trees within a class over a database with multiple classes can be done simultaneously; only small changes to the original mining algorithms are needed.

The next step is to select from all frequent patterns within a class, those patterns that are good descriptors of the class. This is done by means of determining the confidence $P(c_i|T)$ of a rule $T \rightarrow c_i$, that is the probability of a class given the patterns. If this confidence is greater than 0.5, then T is regarded as a good discriminator for class c_i. Note that we did not optimize for the 'optimal' confidence value, nor did we use other measures such as the lift of a rule. The reason for this is, that our goal is to compare different tree matching relation by means of classification and not to build the best classifier. In the set of discriminating patterns found, there still might be redundancy between the patterns. This occurs when a pattern T_1 is a supertree of T_2 and $P(c_i|T_1) \leq P(c_i|T_2)$. In order to reduce the risk of overfitting on the training set and to limit the training time of the classification algorithms all redundant patterns were removed.

The resulting set of discriminating patterns are used as binary features in a standard classification algorithm. Each feature indicates the presence or absence of a discriminating pattern in a record. We used decision trees [11] to learn a classification model, more specifically the implementation provided by Borgelt [5]. This implementation uses a top down divide and conquer technique to build the decision tree. At each node in the decision tree, an attribute is selected—in a greedy manner—that best discriminates between the classes. As selection criterion, information gain was used. Additionally, after construction of the decision tree we used confidence level pruning to avoid overfitting. The parameter used with confidence level pruning was set to 50%. Finally, we estimated the area under the ROC curve. This was done with the method described in the work by Hand and Till [8], where an estimate for the area under the ROC curve is given that can be used for problems with multiple classes.

5 Experiments

In this section we describe the experiments that we performed to answer the research questions stated in section 3. For the two datasets, all minimum support thresholds used and every mining algorithm considered, we computed the following measures:

1. runtime in seconds: the time needed to compute all frequent patterns.
2. # of trees: the number of frequent trees.
3. # of features: the number of non-redundant discriminating patterns.
4. # of nodes in the decision tree: the number of nodes in the decision tree, this is used as a measure for the complexity of the classification model.
5. Accuracy: the ratio of the number of correctly classified documents and the total number of documents in the test set.
6. AUC : area under the ROC curve; this is a measure that compares the classification model with a classifier that has random performance [8]. This measure was also computed on the test set.

All estimates for the different measures were obtained using ten-fold cross-validation. In order to determine whether the difference in estimated area under the ROC curve and the accuracy measure for the different approaches were significant, the standard error for these two measures was computed.

5.1 Data sets

We performed the experiments on two real datasets namely the CSLOG dataset created and used by Zaki and Aggarwal [15] and the Wikipedia XML dataset [7].

The CSLOG dataset consist of user sessions of the RPI CS website, collected over a period of three weeks. Each user session consists of a graph and contained the websites a user visited on the RPI CS domain. These graphs were transformed to trees by only enabling forward edges starting from the root node. The goal of the classification task is to discriminate between users who come from the edu domain and users from another domain, based upon the user's browsing behavior. In total there are $23,011$ trees in the CSLOG dataset, where the average tree size is 8.02. This dataset does not contain any attributes (in the processed version). The relative frequency of the classes is 23.57% for the class that represents the edu domain and 76.43% for the other class.

The Wikipedia XML dataset was provided for the document mining track at INEX 2006. The collection consists of $150,094$ XML documents with 60 different class labels. The collection was constructed by converting web pages from the Wikipedia project to XML documents. The class labels correspond to the Wikipedia portal categories of the web pages. For the document mining track a subset of the total Wikipedia collection was selected such that each document belonged to exactly one class. In our experiments we only used the structure and the attributes of the XML documents. The algorithms that use the embedded subtree relation, were not able to run with the minimum support threshold set to 2% on a server having 4GB of main memory available. The reason for this is described in [6]: in the worst case, for the embedded subtree mining algorithm [13] the scopelist size is exponential in the size of the data tree. In order to include the tree mining algorithms that use the embedded subtree relation into the comparision, we selected only those documents where the number of nodes in the tree was less than 20. Compared with the average size of a tree in the whole collection (161.35) it is a drastic reduction. On the other hand compared with the average tree size of the most commonly used tree-structured dataset (8.02) it is still quite large. Additionally, we only used documents that belong to classes that contained more than 400 documents. The resulting reduced dataset consists of $24,222$ documents that are distributed over 13 classes, where the largest class contains about 22% of the documents and the smallest class about 3%. The average size of tree in the database equals 13.95 and the average number of attribute value pairs attached to a node equals 0.56. For the algorithm where the attribute value pairs are mapped to nodes, the average size of a tree in the database equals 21.73.

5.2 Result and analysis on the Wikipedia dataset

We start with an experiment to determine if the inclusion of attributes has any impact on the quality of the classifiers. Comparing the results of the two methods that include attribute-value pairs into the mining process (ATR and NAIVE) with the algorithm where the attributes were left out (NOATR), the former have a substantially higher predictive performance on the Wikipedia dataset. Both the accuracy and the area under the ROC curve are significantly higher, as shown in figure 2. This result holds regardless of the tree inclusion relation or minimum support value used.

A closer investigation of the patterns used for the ATR and NAIVE classifier, reveals that the main advantage of including attribute values is that these values often describe the document to which a link points. Consider for example the discriminating pattern shown in figure 1. This pattern has very common structural properties, but due to the attribute values it becomes a discriminating pattern. In the example shown, the attribute value points to a picture that was the US Navy Jack for some time. Clearly, the values is a good indicator of its class (“Portal:War/Categories”).

A further remarkable difference between the mining algorithms for the ATR and the NOATR approach, is the number of frequent trees both produce. Although less information is included, the NOATR approach computes far more frequent trees than its counterpart. As a consequence extra computing time is needed for the NOATR mining algorithms. The reason for the increase in frequent trees is that contrary to the ATR approach, the same node labels with different attribute-value pairs are counted as one pattern in the NOATR approach. For example, almost every document in the Wikipedia

```
<figure>
    <image xlink:type="simple" xlink:href="../pictures/USN-Jack.png" </image>
</figure>
```

Fig. 1. A discriminating pattern found on the training set. This pattern, describing class 2879927 (“Portal:War/Categories”) has a support of 48 in its class and of 0 for all other classes.

collection contains a couple of nodes of type “wikipedia link”. These nodes differ only in the attribute-value pairs attached to them. Consequently, in the NOATR approach this will be a highly frequent pattern. This effect is especially noticeable in case the embedded subtree relation is used: the NOATR embedded approach has almost four times as many frequent trees as the ATR embedded approach. On the other hand, for most support values, this is limited to almost two times as many in case the induced subtree relation is used. However, with regard to the number of non-redundant discriminating patterns, the NOATR approach with the induced subtree relation, produces less discriminating patterns than its counterpart with attributes. The ratio between number of discriminating patterns and the number of frequent trees is between $1/18$ and $1/38$ for the ATR approach, while for the NOATR approach the ratio is between $1/58$ and $1/83$. For the embedded subtree relation these ratios are respectively between $1/38$ and $1/53$ and between $1/107$ and $1/149$.

The difference in predictive performance between the classifiers produced with the induced and embedded tree inclusion relation, for the ATR approach is very small, both in terms of accuracy and area under the curve. Although the number of frequent trees and the number of non-redundant discriminating patterns is respectively about two and one and a half times higher for the embedded subtree relation, the complexity of the classification model (measured by the number of nodes in the decision tree) is slightly lower for most cases. For the NOATR classifier, the accuracy and area under the curve estimates achieved are slightly worse for the embedded subtree relation than for the induced subtree relation. Furthermore, the classifier based on the embedded subtrees is by far the most complex classification model for the lower support values.

The patterns produced by the NAIVE mining algorithm are a superset of both the patterns computed with the ATR approach and those computed with the NOATR approach. But, contrary to expectation the results obtained in terms of accuracy are for all but one setting considerably below the results obtained via the ATR approach. And with respect to the area under the curve measure, the performance of the NAIVE approach is for all settings significantly below the performance of the ATR approach. We can not give a complete explanation for this, although with the embedded subtree relation, the classification model is more complex than the classification models based on patterns computed with the ATR approach. This might be an indication for overfitting, however this observation does not hold for the induced subtree relation. Notice, that the features produced by the NAIVE approach are not necessarily a superset of the features produced by the ATR approach. This is because of redundancy pruning.

Furthermore, worth noticing are the number of frequent trees the NAIVE approach produces. Especially in the case that the embedded subtree relation is used, the number of frequent trees and the time needed to compute these are extremely high compared

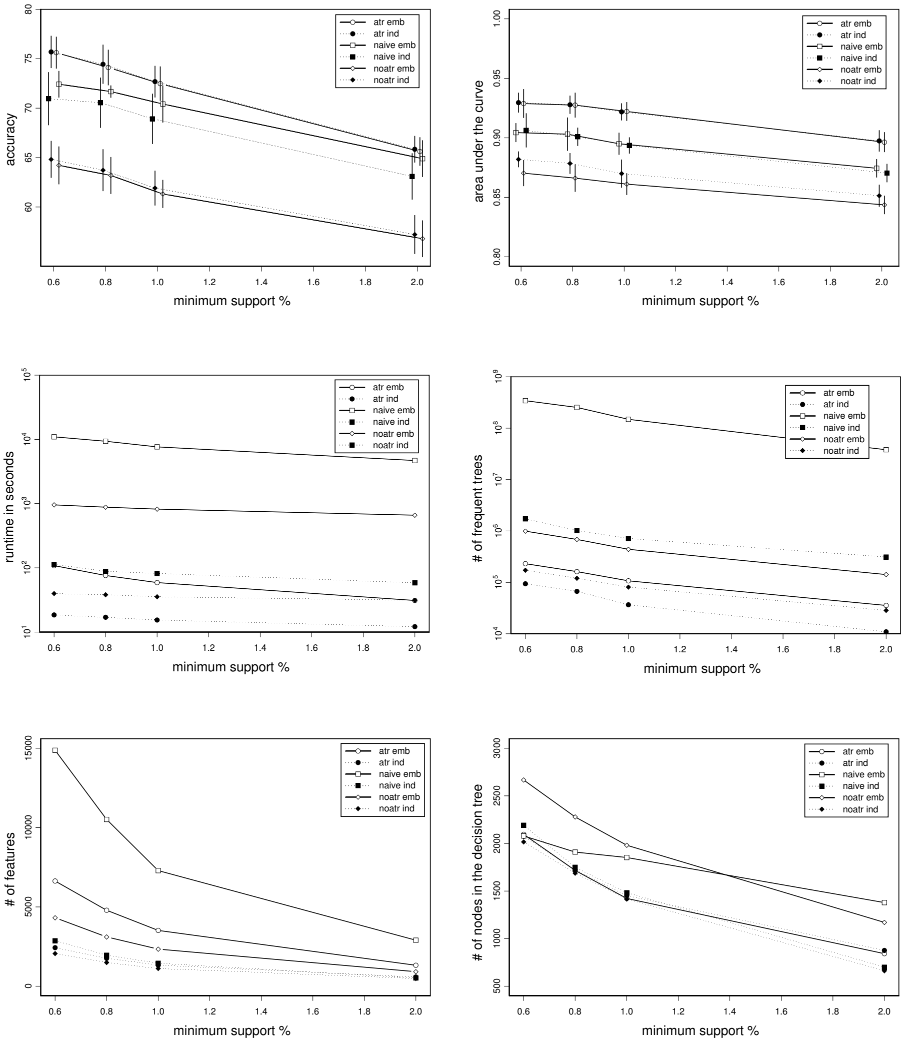

Fig. 2. Results obtained on the Wikipedia dataset. From left to right, top to bottom: accuracy estimates, estimates for the area under the curve, runtime in seconds (log scale), number of frequent trees (log scale), number of non-redundant discriminating patterns, and the number of nodes used in the decision tree. In the two plots on top (accuracy estimates and AUC estimates), for each estimated value we also show $\text{value} \pm 2\text{SE}$, this is shown as vertical bars going through the points. Furthermore, the points in these two plots, are shifted a bit to the right or left, such that the different points and vertical bars could be distinguished.The different lines indicate different combinations of the tree inclusion relation and attribute handling. For example, atr emb denotes the embedded tree inclusion relation with ATR attribute handling.

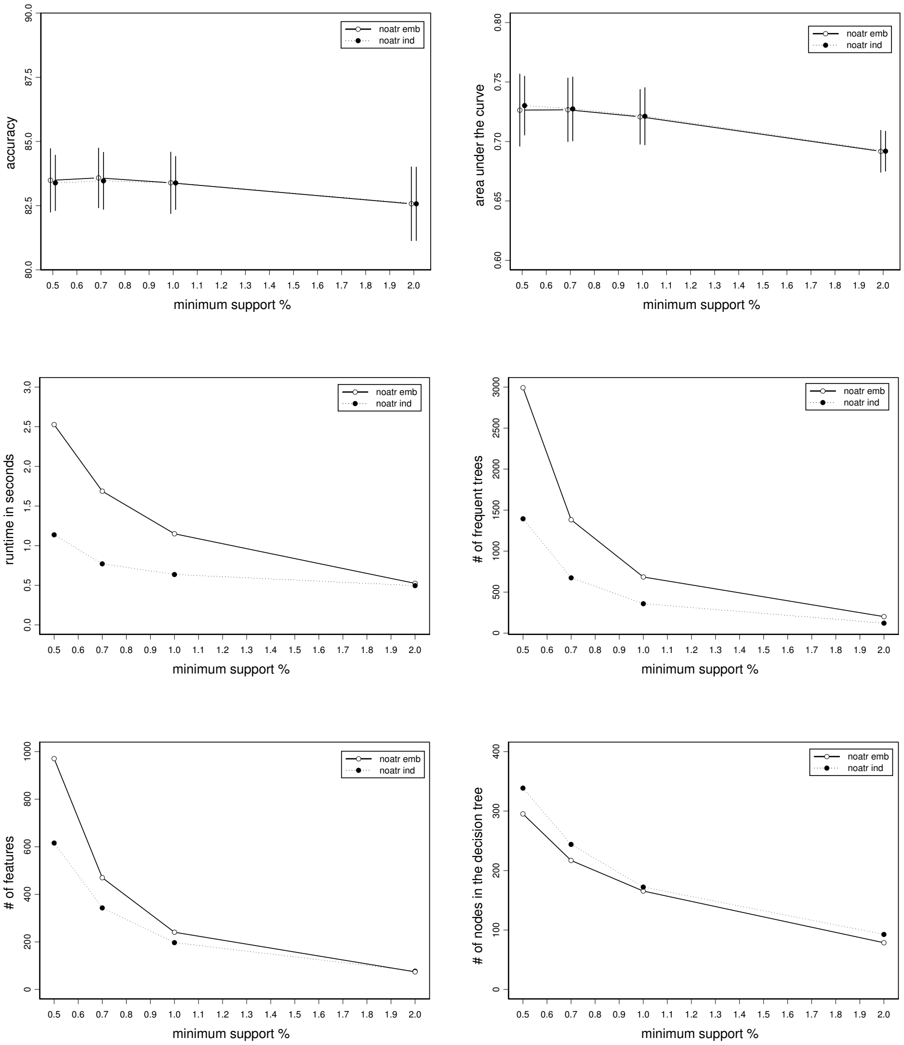

Fig. 3. Results obtained on the Weblog dataset. From left to right, top to bottom: accuracy estimates, estimates for the area under the curve, runtime in seconds (log scale), number of frequent trees (log scale), number of non-redundant discriminating patterns, and the number of nodes used in the decision tree. In the two plots on top (accuracy estimates and AUC estimates), for each estimated value we also shown $\text{value} \pm 2\text{SE}$, this is shown as vertical bars going through the points. Furthermore, the points in these two plots, are shifted a bit to the right or left, such that the different points and vertical bars could be distinguished.

to the other approaches. Furthermore, the ratio between the number of frequent trees and the number of non-redundant discriminating patterns ranges between $1/491$ and $1/595$ for the induced tree inclusion relation and between $1/13146$ and $1/24109$ for the embedded inclusion relation. In comparison with both the ratios of the ATR and the NOATR approach, the NAIVE approach proportionally spends a great effort in computing patterns that will be discarded in a later stage. With respect to AUC, the difference between the induced and embedded subtree relation for the NAIVE approach is negligible. However in terms of accuracy, the embedded subtree relation performs for better than the induced subtree relation.

For all performed experiments on the Wikipedia data set it holds that the difference in predictive performance between the induced and embedded tree inclusion relation is small, and well within the range of the standard error of the estimated accuracy and the are under the ROC curve.

With regard to the run time we earlier noticed that the NAIVE algorithm needs considerably more time than both the ATR and the NOATR approaches, and in turn the NOATR approach consumes some extra time in comparison with the ATR approach. Furthermore, for the embedded tree inclusion relation, considerably more time is needed to compute all frequent trees than with the induced tree inclusion relation. Instead of the total run time, often the average time per tree is derived to compare algorithms. Although this would change the comparison completely, it is for our goal not an interesting measure: we are not interested in the efficiency of an algorithm per se, but in the computation time needed in order to obtain a certain result. Another performance issue is the memory usage of the different approaches. As noted earlier, the Wikipedia dataset had to be reduced such the mining algorithms that uses the embedded subtree relation could be executed with an upper bound of 4GB. For the reduced Wikipedia dataset, the memory usage of all approaches with the induced subtree relation was constant for all minimum support values. These values are respectively 26MB, 46MB and 48MB for the NOATR, ATR and NAIVE approach. For the embedded subtree relation we only measured the worst case, i.e. the case with the lowest minimum support value. For this case the different approaches (NOATR, ATR, NAIVE) used respectively 339MB, 211MB and 520MB. The difference in memory usage is caused by the fact that the only known implementations of the embedded subtree mining algorithm, need to store for every node in the pattern tree a reference to corresponding values in the database. Hence, the larger the pattern tree becomes, the more values (links) need to be stored. This is also the reason why the embedded NAIVE approach uses more main memory than the other embedded approaches: the NAIVE approach has larger frequent patterns. For the induced subtree relation, known implementations need to store only a link to all references in the database from the rightmost node of the pattern tree. Hence, the memory requirement does not depend on the size of the frequent tree but only on the size of the database.

5.3 Results and analysis on the weblog data

The results of the experiments on the weblog data are displayed in figure 3. In comparison with the Wikipedia dataset, the weblog data is relatively simple: both the number of frequent trees and the number of non-redundant discriminating patterns are in comparison quite modest. Also the fact that there are only two classes makes the classification task less complicated. With the embedded subtree relation, both the number of frequent trees and the number of discriminating patterns produced is larger than with the induced subtree relation. However, the accuracy score is, for the lowest support values used, slightly higher for the former. Also in this case, this somewhat higher accuracy score is gained with slightly less features in the classification model. With regard to the area under the ROC curve, the induced subtree relation performs slightly better for the two lowest minimum support values. The run time needed for both tree inclusion relations on this training set is quite small, but is slightly worse for the embedded tree inclusion relation. Also the memory usage of the algorithms is modest: for respectively the embedded and induced approach with the lowest minimum support value it equals 30MB and 25MB.

6 Discussion and Conclusion

In this paper we experimentally compared frequent tree mining algorithms using different tree inclusion definitions, and different ways of handling attributes. Besides the traditional comparison in terms of run time and memory usage, we also constructed classifiers from frequent patterns produced by the different approaches, and compared these classifiers on predictive performance and model complexity.

The three main conclusions of this work are:

1. The inclusion of attribute-values pairs in the mining process significantly improves the classification result.
2. The ATR approach performs considerably better than the NAIVE approach, both in terms of run time and predictive performance.
3. The embedded tree inclusion relation does not result in better classification performance, nor in simpler classification models and has major computational disadvantages.

The results show that significant performance gain can be obtained by including attribute value-pairs into the mining algorithms. Especially the case where attributes are modeled as properties of a node, such as in the ATR approach, delivers good results. But, also in the case where attribute value-pairs are added as elements to the dataset, the results are far better than in case the attributes are ignored.

How the attribute value-pairs are handled in the mining process, is also of great importance for both the predictive performance as well as the runtime. The results show that for all but one minimum support value used, both the accuracy and the area under the ROC curve are considerably better for the ATR approach than for the NAIVE approach. Furthermore, the NAIVE approach requires substantially extra computation time. We conclude from this that, the constraint of “at least one attribute per node” makes sense from both a computational and a usability point of view.

The experiments show no significant difference in performance between the induced an embedded subtree relation. Contrary to the common expectation, the embedded tree inclusion relation often results in a slightly lower predictive performance than the induced subtree relation. Furthermore, a major practical limitation is the excessive memory usage of currently known embedded tree mining algorithms. Even if this implementation issue is solved, then still the embedded approach has as major disadvantage that more computation time is needed, which is spent on patterns that will be thrown away in a later stage.

Although, it is of great advantage to construct less complex classification models that preserve equivalent predictability, the experiments with the embedded tree inclusion relation, did not result in less complex classification models. Concluding, the addition of “hidden patterns” did, for the examined datasets, not result in better predictive performance.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499, 1994.
2. T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In *SIAM Symposium on Discrete Algorithms*, 2002.
3. T. Asai, H. Arimura, T. Uno, and S. Nakano. Discovering frequent substructures in large unordered trees. In *Discovery Science*, pages 47–61, 2003.
4. R. Bathoorn, A.Koopman, and A. Siebes. Reducing the frequent pattern set. In *Workshops Proceedings of ICDM 2006*, pages 55–59, 2006.
5. C. Borgelt. A decision tree plug-in for dataengine. In *Proc. 6th European Congress on Intelligent Techniques and Soft Computing*, 1998.
6. Y. Chi, R. Muntz, S. Nijssen, and J. Kok. Frequent subtree mining - an overview. *Fundamenta Informaticae.*, 66(1-2):161–198, 2005.
7. L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 2006.
8. D. J. Hand and R. J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, 2001.
9. J. De Knijf. FAT-miner: Mining frequent attribute trees. In *SAC ’07: Proceedings of the 2007 ACM symposium on Applied computing*, 2007.
10. S. Nijssen and J.N. Kok. Efficient discovery of frequent unordered trees. In *In Proceedings of the first International Workshop on Mining Graphs, Trees and Sequences (MGTS2003)*, pages 55–64, 2003.
11. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
12. A. Termier, M. Rousset, M. Sebag, K. Ohara, T. Washio, and H. Motoda. Efficient mining of high branching factor attribute trees. In *IEE International Conference on Data Mining*, pages 785–788, 2005.
13. M. J. Zaki. Efficiently mining frequent trees in a forest. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2002.
14. M. J. Zaki. Efficiently mining frequent embedded unordered trees. *Fundamenta Informaticae.*, 66(1-2):33–52, 2005.
15. M. J. Zaki and C. C. Aggarwal. Xrules: an effective structural classifier for XML data. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 316–325, 2003.