Mining semantic rules for optimizing transport assignments in hospitals

Pieter Bonte, Femke Ongenae, Eveline Hoogstoel, and Filip De Turck

Ghent University - iMinds, iGent-Toren, Technologiepark-Zwijnaarde 15, 9052 Gent, Belgium
Pieter.Bonte@intec.ugent.be

Abstract Healthcare is under high financial pressure and hospitals struggle to balance budgets while maintaining quality. In the AORTA project a semantic platform is being developed to optimize transport task scheduling and execution in hospitals by providing a dynamic scheduler with an up-to-date view about the current context gathered by smart devices. This paper details the self-learning module that combines semantic web technologies with association rule mining to learn the causes of late transports.

1 Introduction

Due to the high financial pressure on healthcare, hospitals struggle to balance budgets while maintaining quality. Hence, they are investigating ways to optimize care delivery. One area of interest is the organization of logistic services, which may account for more than 30% of hospital costs [3]. There are huge opportunities in terms of gain in efficiency and cost reduction for transport tasks of patients and equipment. These tasks are scheduled in order to bring patients to various locations in the hospital at the right time, e.g., transporting them to and from radiology and delivering the needed equipment to perform the various medical tasks.

In the AORTA project [8] an intelligent system is being built that assigns the most suitable staff member to a transport based on the available information about the context, staff, patient and requested transport tasks. The current context is gathered by intelligent sensors in the environment and the staff are being equipped with wearables to track their location. Additional background knowledge is gathered from the software and databases already running at the hospital. This information is consolidated by a semantic context layer. A dynamic scheduler receives the transport requests. Based on the context information in the context layer, it constructs an optimal schedule such that all the requests can be handled in a timely manner with an optimal use of resources.

In the context layer, a lot of historical information is thus gathered about requested transports, their context, how they were scheduled and whether they were executed in time. In this paper, a self-learning module is presented that mines this semantic data to give insights into the causes of transports that arrived too late. For example, the module could learn that certain transports during the visiting hours on Friday are often late and more time should be reserved for them. The incorporation of the knowledge modeled in the ontology, allows to learn more accurate and contextualized rules.

2 Related Work

Yu et al. [10] details the importance of using semantic web technologies and data mining techniques in healthcare. The most prevalent methods to learn rules from semantic data are association rule mining (ARM) [7] and Inductive Logic Programming (ILP) [5].

ILP combines inductive machine learning and logic programming. ILP is able to learn rules as OWL-axioms and fully exploits the semantics describing the data. In previous research [2], we have combined ILP with semantic clustering to derive the causes for late transports. However, ILP is very computational intensive, while statistical relational learning methods, such as ARM, tend to be more scalable. Moreover, ILP methods are often inferior when it comes to the generation of highly axiomatized ontologies [9]. Therefore, in this paper, we investigate how ARM can be applied on the semantic transport data to mine it for interesting patterns.

ARM was originally developed to discover hidden knowledge from transactional data, such as relational databases. A transaction is an observation of the co-occurrence of a set of items. ARM can be applied to semantic data by converting it to a set to transactions. Nebot, et al. [7] designed a methodology to perform this conversion. In this research, we further optimize this method and study its application to extract patterns from transport data.

3 Association rule mining: concepts & definitions

$I = \{i_1, i_2, \ldots, i_m\}$ is defined as a set of *m items* and $D = \{t_1, t_2, \ldots, t_n\}$ as a *database of n transactions*, where each transaction is a subset of I. An *itemset* is a subset of items. $supp(X)$ is the *support* of an itemset X. It is the percentage of transactions in the database D that contain X.

An *association rule r* is can then be defined as a rule of the form $X \Rightarrow Y$ where X and Y are non-empty subsets of I, and $X \cap Y = \emptyset$. X is called the *antecedent* of r and Y is the *consequent* of r. The support and confidence of a rule are respectively denoted as $supp(r) = P(X \cup Y)$ and $conf(r) = P(Y|X)$.

Mining association rules is the process of finding all association rules with a support and confidence greater than a particular threshold defined by the user [1]. This mining process can be divided in two phases. First, frequent itemsets of the transactions have to be computed according to the minimum support threshold. Second, rules are generated from these frequent itemsets with respect to the minimum confidence threshold.

4 Self-learning module

4.1 Overview

The processing steps of the self-learning module are shown in Figure 1. In the pre-processing step, the data is split in two clusters, namely the positive and negative examples. In our use case, the first contains the transports that arrived too late, while the latter encompasses the timely ones. Feature selection extracts the features from the ontology that should be used by the ARM algorithm to learn. The transaction extractor

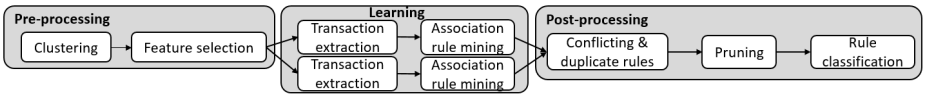

Figure 1. Overview of the various processing steps of the self-learning module

converts the data captured in the ontology for each cluster to transactions that can be used as input for ARM algorithms. ARM is run on each cluster of transactions and outputs the learned rules. In the post-processing step, the rules learned for each cluster are compared. Conflicting rules, i.e., rules that appear in both clusters with the same antecedent but a conflicting consequent, are removed. Finally, the large number of learned rules is pruned and filtered to derive the most interesting ones.

4.2 Pre-processing: clustering and feature selection

Mining association rules about late tasks on the whole data set would require a very low value for minimum support, as these tasks are a minority of all tasks. However, due to the low support value, a lot of irrelevant rules will be learned. For example, as a wheelchair is the most used transport type and it is thus very probably that some of these tasks are late, a rule will be learned that a late transport occurs when a wheelchair is used. To resolve this, the clustering step splits the data into positive and negative examples.

The transaction extractor and ARM need as input a set of selected features. In the original algorithm [7], these are indicated by the user. Since these features heavily influence the type of learned rules and we focus on learning causes for late transports that are not obvious to the end users, a feature selection algorithm is used instead. As the used transport ontology[1] is limited in scope, a basic feature selection method was used. First, all possible attributes of the ontology, i.e., all the concepts and data type properties that are (in)directly linked to the concept we want to learn about, are added to the feature set. Then the variance of each attribute is determined, i.e., the number of different values for the data type property or the number of distinct individuals for the classes. All attributes with zero variance are removed from the feature set. More complex feature selection techniques could easily be plugged into the self-learning module.

4.3 Transaction extractor and association rule mining

The methodology that was used to convert the data captured in the ontology to transactions that can be used as input for ARM algorithms is based on the methodology described in Nebot, et al. [7]. This method was further optimized such that the semantics of the ontology are taken into account more during the learning.

The extraction algorithm is defined by three parameters:

- **the target**: this is the concept to which all selected features must be related.
- **the context**: this is the concept that should be analyzed by the mining algorithm and on which the transactions are built.

[1] More information about used ontology: http://pbonte.github.io/ontology/aorta

– **the features**: these are the set of features that were extracted by the feature selection algorithm explained in Section 4.2.

The transaction extractor takes the ontology as input and constructs transactions for each instance of the context concept. A transaction is composed of features that belong to the same context instance. The algorithm consists of four steps:

– **Step 1, generating composition triples**: A composition triple connects each instance $i_T \in \{\text{target instances}\}$ to instances $i' \in \{\text{features}\}$ that are connected through a path p. The path p must contain one instance i_C belonging to some concept of the context.
– **Step 2, grouping composition triples**: The composition triples with the same target instance and the same path sliced at the position of the context instance, are grouped together. These paths contain all information about that particular context instance.
– **Step 3, generating transactions**: For each group of composition triples, a transaction is constructed. The algorithm iterates over the triples in the same group and for each triple an item is added to the transaction. The name and value of the item depends on the path and the type of the feature in the triple.
– **Step 4, adding items to transactions based on ontology hierarchy information**: As can be noted from the previous steps, the original algorithm only takes limited semantic information into account. Therefore, we implemented an optimization that is able to learn more general rules by generating extra items in the transactions that encode the hierarchy of the ontology.

The result of the transaction extractor is a collection of transactions that can be fed to traditional ARM algorithms as explained in Section 3.

4.4 Post-processing: conflicting & duplicate rules, pruning and rule classification

After the learning step, two sets of association rules are obtained, one for the late transports and one for the transports that arrived on time. First, conflicting rules, i.e., rules that appear in both sets with an equal antecedent but an opposite consequent, are deleted. What can be considered as conflicting consequents, needs to be specified by the user as input to the post-processing step. Duplicate rules, i.e., rules that appear in both sets, are deleted as they cannot be used to discern late transports from the timely ones.

Generally, a lot of rules will be generated, which might make it difficult for end users to discern the interesting findings. Rule templates [4] can be used to prune the interesting rules. A rule is matching a pattern defined by a template, if the rule can be considered as an instance of the pattern.

First, restrictive rule templates can be defined. These templates express knowledge that the end user already knows and are thus uninteresting to learn. Second, inclusive rule templates can be defined, that express knowledge that is interesting to the end users. These can also be defined by the end users. Based on these inclusive rule templates, the rules are classified into four groups [6]:

– **Conforming rules**: these rules match at least one inclusive rule template
– **Unexpected antecedent rules**: the antecedents of these rules do not match on the antecedents of the inclusive rule templates, but the consequent does match on the consequent of at least one of the inclusive rule templates

– **Unexpected consequent rules**: the consequent of these rules do not match on the consequent of the inclusive rule templates, but the antecedents do match on at least one of the inclusive rule templates
– **Unexpected rules**: these rules do not match at all with the inclusive rule templates

5 Evaluation

To thoroughly evaluate the performance of the designed self-learning module, an artificial dataset was generated based on the characteristics of a real hospital and statistics derived from transport logging data received from two hospitals. The generated dataset contains 23,000 transports, which is the average number of transports that occurred over a period of three months in the studied hospitals. The following rules where incorporated in the dataset, allowing to correctly evaluate the learning capabilities of the algorithm:

$$\texttt{BedWithBedmover} => \texttt{ArrivalTooLate} = \texttt{True} \quad (1)$$

$$\texttt{Day} = \text{"}\texttt{Monday}\text{"} => \texttt{ArrivalTooLate} = \texttt{True} \quad (2)$$

$$\texttt{BeganTooLate} = \texttt{True} => \texttt{ArrivalTooLate} = \texttt{True} \quad (3)$$

$$\texttt{PatientNotReadyError} => \texttt{ArrivalTooLate} = \texttt{True} \quad (4)$$

To identify the performance of the designed algorithms, the number of *relevant* and *irrelevant* learned rules are calculated. The relevant rules are defined as the conforming rules, while the irrelevant rules are comprised of the unexpected antecedent or consequent rules and the totally unexpected rules. The learning performance is calculated based on the precision and recall of the algorithm. The precision is calculated as $\frac{TP}{TP+FP}$ and the recall as $\frac{TP}{TP+FN}$. With the true positives (TP) the number of learned rules that are explicitly covered in the dataset, the false positives (FP) the number of learned rules that are not explicitly covered in the dataset and the false negatives (FN) the number of rules that are covered in the dataset but not learned by the algorithm.

Table 1 depicts the results of running the self-learning module with an increasing amount of processing steps enabled. In test 1, only the clustering step of the preprocessing phase, the complete learning phase except the optimization of the transaction extractor, and the conflicting and duplicate rules step of the post-processing phase are included. The features for the learning phase are manually selected in such a way that all the included rules can be learned. In test 2, the feature selection step is included. In test 3, the optimization of the transaction extractor, which makes sure that the learning algorithm takes the hierarchy of the ontology into account, is added. Finally in test 4, the pruning and rule classification steps are enabled. This allows to assess the impact of all these optimizations that we added to the ARM algorithm on semantic data to optimize it for our use case.

6 Discussion

The results show that by splitting up the dataset in two clusters and comparing the learning results, a lot of irrelevant rules can be removed, which results in a high precision and recall. The recall is very dependent on the set of manually selected features. A poorly

Table 1. Evaluation of the different processing steps of the self-learning module. The number of conflicting (C), duplicate (D), pruned (P), relevant (Rel.) and irrelevant (Irrel.) rules are shown. The TP are shown between brackets.

			Before post-processing		**post-processing**			**After post-processing**		**Evaluation**	
	Removed features	Frequent itemsets	Rel. rules	Irrel. rules	C rules	D rules	P rules	Rel. rules	Irrel. rules	Precision	Recall
test 1	-	31	12 (3)	23	7	14	-	3 (3)	9	100%	75%
test 2	13 of 79	73	36 (3)	191	33	93	-	3 (3)	98	100%	75 %
test 3	14 of 82	87	43 (4)	209	37	93	-	6 (4)	116	66%	100%
test 4	14 of 82	87	43 (4)	209	37	93	6	6 (4)	110	66%	100%

chosen set of features would reduce the recall. Adding the feature selection step, resolves this issue. The automatic feature selection remove 13 features with zero variance and achieves the same precision and recall as the manual feature selection.

Test 3 illustrates that adding the optimization to the translation extraction, allows the algorithm to take the ontology hierarchy into account and learn more general rules. In this test, Rule (4) is also learned. As such, a perfect score for the recall is achieved. However, two additional relevant rules were also learned, that are more general variants of Rule (1). Therefore, the precision drops.

Test 4 illustrates that the pruning step is able to remove some of the irrelevant rules without affecting the relevant ones. For the used ontology, 142 restrictive rule templates were generated. 6 rules matched at least one of these templates.

Finally, it can be noted that by specifying the inclusive rule templates, it is very easy to detect and visualize to end users, which of the learned rules are relevant.

References

1. Agrawal, R., et al.: Mining Association in Large Databases. IBM Almaden Research Center pp. 207–216 (1993)
2. Bonte, P., et al.: Learning semantic rules for intelligent transport scheduling in hospitals. In: Proc. of ESWC (2016)
3. Hastreiter, S., et al.: Benchmarking logistics services in German hospitals: A research status quo. In: Proc. of ICSSSM. pp. 803–808 (2013)
4. Klemettinen, M., et al.: Finding interesting rules from large sets of discovered association rules. In: Proc. of CIKM. pp. 401–407 (1994)
5. Konstantopoulos, S., Charalambidis, A.: Formulating Description Logic learning as an Inductive Logic Programming task. In: Proc. of FUZZ (2010)
6. Marinica, C., Guillet, F.: Knowledge-based interactive postmining of association rules using ontologies. TKDE 22(6), 784–797 (2010)
7. Nebot, V., Berlanga, R.: Finding association rules in semantic web data. Knowledge-Based Systems 25(1), 51–62 (2012)
8. Ongenae, F., et al.: Semantic context consolidation and rule learning for optimized transport assignments in hospitals. In: Proc. of ESWC (2016)
9. Völker, J., Niepert, M.: Statistical schema induction. In: Proc. of ESWC. pp. 124–138 (2011)
10. Yu, W.D., Jonnalagadda, S.R.: Semantic web and mining in healthcare. In: Proc. of HEALTHCOM. pp. 198–201 (2006)