An explicitly modeled algorithm for mining frequent item sets in MDE settings

T. Leys[a], I. Dávid[a,b], C.G. Gomes[a], H. Vangheluwe[a,b,c]

[a] *University of Antwerp, Belgium*

[b] *Flanders Make, Belgium*

[c] *McGill University, Montréal, Canada*

Abstract

Frequent itemset mining is a well known algorithm within data mining applications. Existing implementations of the algorithm depend on a textual representation of the transactions. When dealing with a formalism that has a visual concrete syntax, finding a meaningful textual representation can be challenging. In this paper we will make a visual representation of a transaction using mde techniques. We will also show an explicitly modeled algorithm for frequent itemset mining.

Keywords: MDE, Frequent Itemset Mining, Explicit Modeling

1. Introduction

Recommender systems has become a ubiquitous concept in the world of computer science. By analyzing past behavior, a recommender system tries to recommend certain items to a user. A typical applications of recommender systems can be found in e-commerce sites like amazon.com. A user will be given a set of items that he is likely to be interested in. Andrei Dyck et al. [3] have mentioned the fact that modeling environments might benefit from recommender systems, however current research in this field is limited.

Email addresses: `tim.leys@student.uantwerpen.be` (T. Leys), `istvan.david@uantwerpen.be` (I. Dávid), `claudio.goncalvesgomes@uantwerpen.be` (C.G. Gomes), `hans.vangheluwe@uantwerpen.be` (H. Vangheluwe)

A widely used technique in recommender systems is frequent dataset mining. Given a database of transaction, the technique tries to identify which items appear frequently together in a transaction. A transaction is a set of items, in the context of e-commerce a transaction can be interpreted as a set of items a user buys together. Currently, many algorithms have been developed to deal with this problem, but they all operate on textual transactions. In model driven engineering however, formalisms often have a visual concrete syntax. To use the existing algorithms we would have to transform models into a textual representation. It is hard enough to reason about the importance of visual elements for the algorithm, that transforming those features into a textual representation (and back) adds unnecessary complexion. In this research we will come up with a meaningful visual representation of transactions that can be used as input to an algorithm.

Thomas Kühne et al. [6] have stated the importance of explicitly modeling models as well as model transformations. In this paper we will explicitly model the frequent itemset mining algorithm. Explicitly modeling the algorithm has some advantages. The specification is not hidden away in code, the specification can be altered on the fly and its easier to reason about it. We will show that our algorithm will be able to deal with the visual representation of transactions.

We will also apply the algorithm to a motivating example situation of a tile factory. The example will show the possible benefits of applying data mining techniques to modeling formalisms and dsl's.

2. Related work

2.1. Mining frequent itemsets

Frequent itemset mining is a very well known problem within datamining and plays an essential role when mining associations, frequent patterns, causal structures [7], etc. Many algorithms have been proposed, such as the apriori algorithm [1], eclat [9] and the fp-growth algorithm [5]. The idea is that, given a database containing itemsets, the algoritm outputs the support of every possible superset of items. The support of an itemset indicates that a certain combination of items appears frequently in the database. The support can be used to compute the confidence of association rules (these rules are explained in section *2.2*).

These algorithms work on batches of data. For this project it would be interesting to have an algorithm that is specialized for an incremental database. The algorithm that uses CATS-Trees [2] seems a very good candidate, since it is an extension on the FP-Tree, allowing a compressed representation of the itemsets.

2.2. Association rules

Association rules are rules of the form $[x, y] \rightarrow [a]$[8]. The meaning of such a rule is that if x and y both appear in the same itemlist, it is very likely that a will also be in this itemset. If we can find this kind of associations regarding the elements in a modelling formalism, we can make an estimation of how likely it is that a user will use that element, given the current model.

2.3. RETE Networks

The RETE network, as described by Forgy[4] is an elegant implementation of a many-object/many-pattern matching algorithm. The idea behind it is to limit iterating over the objects as well as the iterating over patterns. The first idea is to store with each pattern a list of objects that it matches. This way, when a production is performed, the algorithm only has to look at the sets of the pattern, rather than scanning the whole database. When a new item enters the set of objects, or an old one leaves it, the patterns that match it will update their sets. To compute those patterns, the match algorithm will not iterate over every pattern. Based on the patterns, the algorithm will build a network of nodes, each containing a test for a single attribute of the object. When two patterns have the same test on an attribute, the node will be shared among the two patterns, meaning that the test would only be performed once.

The RETE network might be a candidate to address the problems raised in the motivating example.

3. Motivating example

To demonstrate the usefulness of the study, an example situation will be used. As example, we will use a tile manufacturer/retailer that is interested in a visual formalism for modeling a clients floor and how he would place the tiles. This model will allow the client to make a more precise estimation

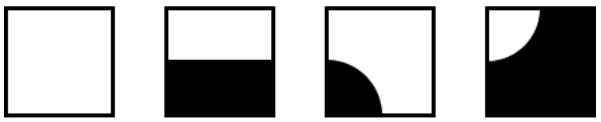

Figure 1: Concrete syntax of the tile formalism.

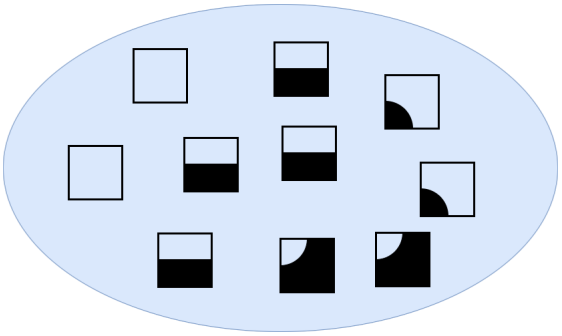

Figure 2: Transaction model of tiles.

on the amount of tiles required and he can immediately see how the pattern of tiles will look on his floor. Figure 1 contains an example of the syntactic elements for tiles.

These models can be mined for useful information. A possible interest might be that we want to mine association rules. These rules would indicate how likely a user would use certain tiles, given a partially completed model. To do this we need to find a way to transform the model into a meaningful transaction. In this case, a transaction should contain the set of all tiles in the model, an example of such a transaction model can be seen in Figure 2.

Another point of interest might be to look for components that are frequently ordered together. This information is very interesting for proactive production of certain components. The model of a transaction from the previous example will not produce the desired output, instead we can make a formalism that models the different components needed to make the tiles and make model transformation to a transaction model as shown in Figure 4.

We can easily make the link between this example and Industry 4.0. Through

Figure 3: Concrete syntax for the component formalism.

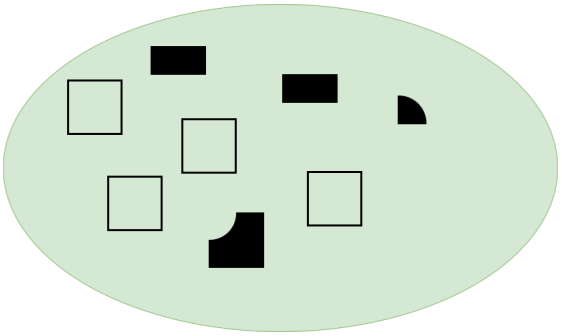

Figure 4: Transaction model of components.

mining the models made in the formalism, we can feed information into the production equipment, such that it can automatically and pro-actively change the production rates of certain components.

We can also see that the problem in the example is related to multi paradigm modeling. We have to deal with two formalisms, both applying to a different level of abstraction.

4. Project

The project will consist out of two parts. The first part is to create a simple formalism for the example that we are going to observe. Next, we will need model transformation to go from a model in the formalism to a meaningful transaction. We will create the formalism and transformations in AToMPM, since we can use autosnapping and containment based associations.

The second part of the project will be to model the algorithm. It might be useful to model the algorithm using the eclipse modeling framework. This framework has been proven to be highly efficient, regarding execution time.

If time allows it, we can do a performance study against a similar code based algorithm.

5. Planning

25 December 2017	A working formalism for making tiles in AToMPM.
31 December 2017	Having a working model transformations to go from a model to a transaction model.
10 January 2018	Model of the frequent itemset mining algorithm.
20 January 2018	Interpret results.
29 January 2018	Finish report and presentation.

6. Bibliography

[1] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[2] William Cheung and Osmar R Zaiane. Incremental mining of frequent patterns without candidate generation or support constraint. In *Database Engineering and Applications Symposium, 2003. Proceedings. Seventh International*, pages 111–116. IEEE, 2003.

[3] Andrej Dyck, Andreas Ganser, and Horst Lichter. Model recommenders for command-enabled editors. *MDEBE2013*, 2013.

[4] Charles L Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial intelligence*, 19(1):17–37, 1982.

[5] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM, 2000.

[6] Thomas Kühne, Gergely Mezei, Eugene Syriani, Hans Vangheluwe, and Manuel Wimmer. Explicit transformation modeling. In *International Conference on Model Driven Engineering Languages and Systems*, pages 240–255. Springer, 2009.

[7] Craig Silverstein, Sergey Brin, Rajeev Motwani, and Jeff Ullman. Scalable techniques for mining causal structures. *Data Mining and Knowledge Discovery*, 4(2-3):163–192, 2000.

[8] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[9] Mohammed Javeed Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.