Transforming Association Rules to Business Rules: EasyMiner meets Drools

Stanislav Vojíř, Tomáš Kliegr, Andrej Hazucha, Radek Škrabal, Milan Šimůnek

Department of Information and Knowledge Engineering, University of Economics, Nám. Winstona Churchilla 4, Prague 3, 130 67, Czech Republic
{stanislav.vojir|tomas.kliegr|andrej.hazucha|xskrr06|simunek}@vse.cz

Abstract. EasyMiner (`easyminer.eu`) is a web-based association rule mining software based on the LISp-Miner system. This paper presents a proof-of-concept workflow for learning business rules with EasyMiner from transactional data. The approved rules are exported to the Drools business rules engine in the DRL format. The main focus is the transformation of GUHA association rules to DRL.

1 Introduction

The EasyMiner association rule mining system discovers rules from a table of objects. The system outputs all rules which hold in the given dataset in a certain predefined statistical sense. An example of a rule is $\ulcorner$Amount=$\langle 100.000; 200.000)$ $\wedge$ `District=Prague` $\rightarrow_{0.7,100}$ `Status=A`$\urcorner$. Such a rule is learnt from a table (data matrix), where each row corresponds to one client of a bank, and it contains at least the following data: amount borrowed, district of the customer and loan status. Rule confidence 0.7 denotes that in this table, it is true that for 70% of clients from Prague who borrowed 100 to 200 thousand Czech crowns, the loan was A-grade. The *support* of the rule is 100, which means that there were at least 100 such clients.

The discovered rules are either exploited in a qualitative way by an expert, or used to perform classification (scoring) of incoming objects (e.g. [7]). With EasyMiner we attempt for a midway between these approaches: expert selects only some of the discovered rules, which are then interpreted as business rules. While the idea of interaction of a domain expert with discovered rules is not new [2], to the best of our knowledge, EasyMiner is the only web-based system which supports the complete cycle: data upload, preprocessing, mining, user interaction with the discovered rules, and export of selected rules to a business rules engine.

This paper is organized as follows. Section 2 describes EasyMiner and its workflow. The syntax of association rules output by EasyMiner is detailed in Section 3. Section 4 describes the transformation of rules to the DRL format. The description of the demo and the access details are listed in Section 5. The paper is concluded with some remarks on the applicability of the described transformation setup to other rule learners and with outlook for further work.

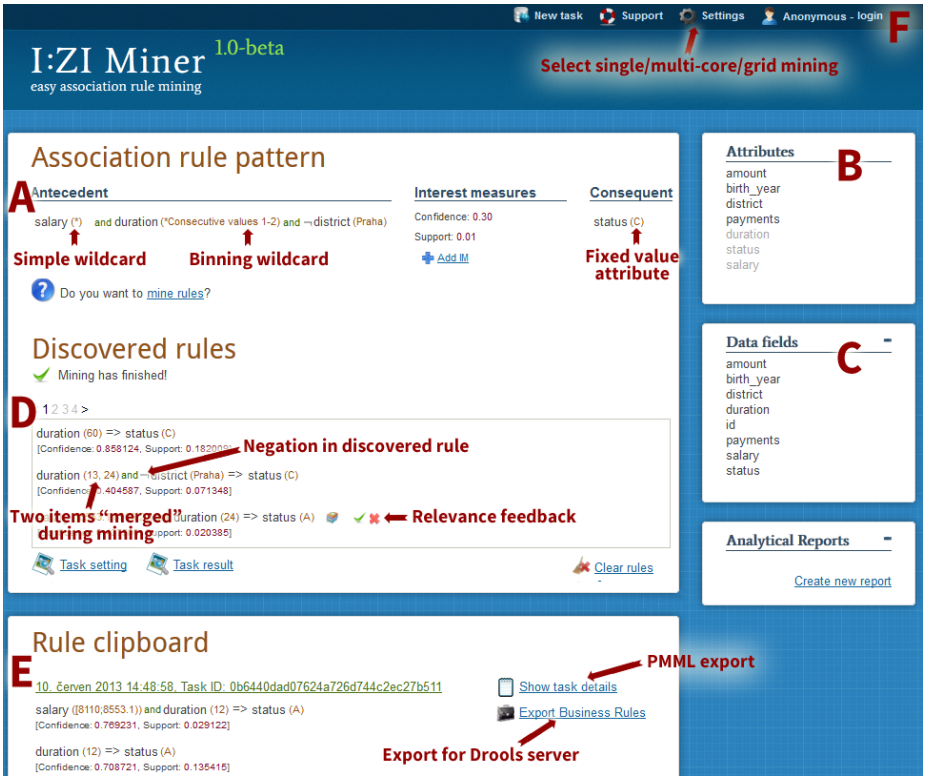

Fig. 1. EasyMiner screenshot

2 EasyMiner

EasyMiner is a sister project of the association rule learning system LISp-Miner (`lispminer.vse.cz`, [10]), which is a desktop/server-based system developed since the mid-1990s. The original paradigm of rule mining in LISp-Miner was that the discovered rules are pieces of knowledge intended for "human consumption". EasyMiner, introduced at ECML'12 [12] as I:ZI Miner,[1] is both an interactive web application, which allows interactive pattern mining, and a web-service layer on top of LISp-Miner.

EasyMiner allows the user to perform the complete association rule mining task and review the discovered rules from an Internet browser.

[1] The first predecesor of EasyMiner called Association Rule Query Designer was introduced in [4]. This system was used for querying mining results stored in a knowledge base, not for performing live mining.

2.1 Data import

The imported data are in tabular form (a CSV file or a MySQL table). For columns with many distinct values it is strongly recommended to perform preprocessing by grouping similar values into a smaller number of *bins*. This can be done either automatically by a built-in heuristic algorithm on data import (numeric fields only), or manually after the mining task is setup. An example of a binning result is replacing all the say 60 distinct values of attribute "age" with just five values such as $\langle 15; 23\rangle, (23; 37\rangle, (37; 49\rangle, (49; 53\rangle, (53; 75\rangle$.

2.2 Defining the Mining task

Once the data are imported, the user is presented with the main EasyMiner screen. The mining task is defined in the *Pattern Pane* (Fig. 1A) by selecting interest measures and placing attributes from the *Attribute Palette* (Fig. 1B) on the left and right side of the rule.

The set of interest measures includes the industry-standard confidence, support and lift measures and about 10 additional measures. All measures can be freely combined. The setting of an interest measure also involves a selection of a threshold value.

By dragging attributes on the left and right side of the rules respectively, the user decides which attributes *might* appear in the rule. For each attribute, it is also possible to define the set of its values considered during mining using the following options:

- fixed value: attribute must use a specific bin as its value if it appears in a rule
- simple wildcard: the system tries all single bins for the attribute value
- dynamic binning wildcard: during mining time, the system creates broader bins by merging bins created in the preprocessing stage into one bin. An example of a dynamically created bin is $\langle 15; 23\rangle \vee (23; 37)$.

It should be noted that while dynamic binning wildcard is convenient, it can significantly increase the computation time. To alleviate this problem, the user can select from several dynamic binning wildcards and thus restrict the size of the hypothesis space (e.g. only consecutive values are attempted to be merged). When the originally created preprocessing does not produce satisfactory results and dynamic binning does not help or is computationally infeasible, the recommended action is to create new attributes by dragging the names of columns from the input data from the *Data Field Palette* (Fig. 1C) to the *Attribute Palette* (Fig. 1B). After dropping the column to the *Attribute Palette*, the user defines custom preprocessing (binning). In this way, the mining task can contain multiple attributes derived (different binning) from the source data field.

Manual binning has also one significant advantage in the business rules context: bins can have user friendly names. Instead of bin $(53; 75\rangle$ (result of automatic binning), the user can create more meaningful bins, e.g. by creating a bin $\langle 60; 75\rangle$ and naming it "senior".

2.3 Mining

Once the user completes the setting of a mining task and clicks on the *mine rules* link, EasyMiner converts all the user settings to a variant of the PMML format [6] and submits the task via a web service to LISp-Miner. Depending on the configuration, defined via the Settings link (Fig. 1F), the task is executed in a single or multi-threaded LISp-Miner instance, or on the grid [11].

The discovered rules are returned to the EasyMiner front-end incrementally, as LISp-Miner progresses through the search space. Real-time results are shown in the *Result Pane* (Fig. 1D).

2.4 User Interaction with Results

The user oversees the discovered rules and tries to select the ones, which he or she thinks would bring value when deployed. The system offers two aids to the user: the strength of the rule and filtering based on a knowledge base.

The strength of the rule is indicated by the value of interest measures which the user selected in the *Pattern Pane*. The values for all discovered rules displayed there meet or exceed the preset thresholds. Generally, the higher the interest measure value above the threshold, the better the rule. Despite this simple "metarule of thumb", the user should understand the semantics of the interest measures. As a future extension of the system, we plan to provide a representation of the rule in a human-friendly textual form, which should lower the requirements on user training (see Sec 6).

Discovered rules can be checked against a knowledge base of stored rules by issuing a *confirmation* or *exception* query [5]. Confirmation query returns rules from knowledge base, which contain in the antecedent only attributes contained in the discovered rule's antecedent, and for each of these attributes, there is at least one overlapping value. The same must apply for the consequent. Exception query returns rules with the same antecedent and a consequent which share at least one attribute, and at least in one of the shared attributes there is no overlap in attribute values.

EasyMiner makes the check of the discovered rules against the knowledge base transparent for the user by embracing a relevance feedback paradigm: if the discovered rule is only a confirmation of a rule in the knowledge base, it is visually suppressed by gray font. In contrast, if the rule is an exception, it is highlighted in red. A green tick, moving the rule to the Rule clipboard, also stores the rule into the knowledge base. The relevance feedback module is a Java application running on top of the XML Berkeley database, which communicates with EasyMiner via a web service.

2.5 Rule Clipboard

The rules confirmed by the user are moved to the *Rule clipboard* (Fig. 1E). The rules in the clipboard are grouped according to the task, in which they were discovered. By clicking on the "Show task details" button, the user is presented

an HTML page with a complete definition of the mining task and the description of the data. Technically, this report is generated with an XSLT transformation from the GUHA AR PMML [6] XML export of the LISp-Miner system, which is available under the *Task result* link in the Result Pane (Fig. 1D).

The *Export Business Rules* link exports the rules in the clipboard for a specific task to the Drools server. For demo purposes, this link shows the DRL serialization of the rules.

3 BR-GUHA Association Rule

In theory, the LISp-Miner system used by EasyMiner mines generic GUHA association rules [9]. The high expressivity of GUHA rules is not suitable for this initial work on the transformation of association rules to business rules. While EasyMiner contains some simplifications in comparison with the full LISp-Miner implementation, the "EasyMiner" rules are still too expressive. In this section, we describe BR-GUHA 0.1, a constrained version of GUHA rules, which is suitable for transformation to Business Rules.

In the formal definition of GUHA rules, antecedent and consequent of the rule are defined in terms of boolean attributes, which are, in turn, defined as conjunction or disjunction of boolean attributes or literals. EasyMiner simplifies this generic recursive structure to a fixed three layer model, which eases the manipulation with the discovered rules:

- Layer 1: Antecedent is a conjunction of *derived boolean attributes*, Consequent is a non-empty conjunction of *derived boolean attributes*,
- Layer 2: A derived boolean attribute is a conjunction or disjunction of *literals*,
- Layer 3: A *literal* is an attribute-value pair or its negation.

Further, it should be noted that:

- *Attribute* refers to the result of preprocessing, not to a field in the original data table,
- *Value* is a bin created during preprocessing, or a dynamically created bin (a disjunction of multiple bins).

By default, EasyMiner (and GUHA) allows the consequent of the rule to have the same rich structure as the antecedent. The consequent of the rule can thus contain for example a disjunction of multiple attributes, or a disjunction of values of one attribute.

In contrast, with Business Rules, a rule needs to have a *definite* outcome. To quote from the Drools documentation: *It is bad practice to use imperative or conditional code in the RHS of a rule; as a rule should be atomic in nature - "when this, then do this", not "when this, maybe do this"*[2]. In BR-GUHA the

[2] `http://docs.jboss.org/drools/release/6.0.0.Beta3/drools-expert-docs/html_single/index.html#d0e7386`

consequent of the rule is constrained to contain a positive literal (negation not allowed). Furthermore, the attribute value must correspond to a single value in the underlying data table (no binning or dynamic binning allowed). No restrictions are made to the antecedent of the rule.

The second important component of an association rule are the interest measures, the 4ft-quantifier in *GUHA* terminology [3, 9]. A 4ft-quantifier is composed from one or more 4ft-partial quantifiers, each associated with one or more quantifier values. While EasyMiner embraces the more commonly used term "interest measure", in other respects it does not impose additional constraints.

In BR-GUHA, we constrain the EasyMiner setup to two interest measures. Only the most commonly used interest measures are supported: *confidence*, *support* and *lift*, all with just one associated value. The first measure must be support, and the second measure is either lift or confidence.

Technically, the constraints described in this section are imposed by not allowing certain features in the mining setup. Most BR-GUHA constraints are readily supported by EasyMiner.[3]

4 Representing EasyMiner Association Rule in DRL

This section describes an initial specification of the conversion procedure of the simplified GUHA rules ("BR GUHA 0.1") to the Drools Rule Language (DRL). In this preliminary work, this specification is done informally, through examples of transformation result for the relevant syntactic features.

4.1 Running Examples

Throughout this section, two example GUHA rules will be used. The first takes up the simple rule from the Introduction, while in the second all the syntactic features are used.

Rule 1
$\ulcorner$Amount=$\langle 100.000; 200.000)$ $\wedge$ `District=Prague` $\rightarrow_{0.7,100}$ `Status=A`$\urcorner$,
where 0.7 is the confidence value and 100 the support.

Rule 2
$\ulcorner$`(Amount=`$\langle 100.000; 200.000)$ $\vee$ `Duration=1year)` $\wedge$ $\neg$`(District=Bruntal)`
$\wedge$ `(Age=[Senior` $\vee$ `Student]` $\vee$ `Payments=`$\langle 5.000; 10.000)$`)`
$\wedge$ `Education=university` $\rightarrow_{0.95,20}$ `Status=B`$\urcorner$,
where 0.95 is the confidence value, and 20 the support.

4.2 Attributes

To comply with the Drools object-oriented principles, each attribute in a rule is transformed to an instance of the Drools `Attribute` class. In the following, we will refer to this instance as `DrlObj`.

[3] With the exception of disabling binning in the preprocessing stage

The names of the attributes in the discovered rules may not necessarily match the names of fields in the underlying data table. Since it is expected that the requests to the business rules engine will use the names of the fields from the underlying data table, rather than the custom names introduced during data preprocessing, the name of the instance is set to the name of data field on which the attribute is based. The same applies to attribute values.

Rule 1 features attribute-value pair `District=Prague`. Assuming that the name of the underlying data field is "district", and the underlying data value "Praha" was renamed during preprocessing to "Prague", the resulting DRL fragment is as follows:

```
DrlObj (name == "district", value == "Praha")
```

4.3 Interest measures

The action of a user confirming the rule and exporting it to the business rule system, strips away the "fuzziness" from the rule, replacing the interest measure with a causal relationship. The original value of interest measures can, however, be used to define the conflict resolution strategy.

Consider object 1 depicted in Table 1.

ID	amount	district	age	duration	payments	education
1	120.000	Praha	63	1year	11.000	university
2	110.000	NA	61	1year	9.000	university

Table 1. Example objects

Both Rule 1 and Rule 2 match this object, however, the consequents of these rules are conflicting, since the status cannot be both A and B.

Drools offers multiple conflict resolution strategies. Interest measure values can be utilized in the *salience* strategy, by setting the salience property of a rule according to the value of lift or confidence interest measures, whatever is used in the rule. Since salience in Drools is an integer, while confidence is a float in the range of $\langle 0; 1\rangle$ and lift is a float in the $\langle 0; inf\rangle$ range, the original value of the interest measure needs to be multiplied by a scaling factor, e.g. 100, before it can be used as salience.

In association rule mining, it can be generally observed that with the increasing specialization of the antecedent, the confidence of a rule rises at the expense of decreasing rule support (as exemplified by Rule 1 and Rule 2). Specific rules are therefore preferred as their consequents are more likely to hold than for a consequent of a conflicting rule with a smaller number of conditions. To this end, the Drools *complexity* conflict resolution strategy, which favours rules with more conditions, should yield similar results as the salience strategy.

It should be noted that the statistical validity of a rule decreases with increasing specificity as each condition filters out some objects that would contribute to the *support* of the rule. However, we suggest not to take support into account during conflict resolution, since the fact that all rules considered have sufficiently high support is ensured by:

- support of the rule exceeding the minimum threshold set by the user during the mining setup,
- the user has explicitly approved the rule by placing it into the rule clipboard.

The use of the complexity strategy rather than the salience strategy also has the advantage that it naturally solves the situation when there are multiple conflicting rules with different interest measures. In this case, a comparison of salience would not make sense: the salience of 70, derived from confidence 0.7, and salience 110, derived from lift value 1.1, are incomparable.

Our preliminary conclusion is that the first approach to handle interest measures in the DRL export is to ignore them, and to use the complexity resolution strategy instead.

In our example, this strategy would favour Rule 2 over Rule 1.

4.4 Binning

The values of attributes are a result of binning. The names of bins can be automatically generated, user-defined, or the same as the values of the underlying fields in the data table.

Since it is expected that the requests to the business rules engine will use values from the underlying data table, rather than the bin names, it is necessary to translate the bin names back to the values of the underlying datafield. In this step, one bin will be replaced by one or multiple values.

The resulting DRL depends on the data type of the attribute (numerical, nominal).

Numerical attributes The bins of the Amount attribute from Example 1 are created on a numeric range.

$$\ulcorner \text{Amount}{=}\langle 100.000; 200.000)\urcorner$$

The result of transformation to DRL:

```
DrlObj(name == "amount", numVal >= 100000, numVal < 200000)
```

Nominal attributes The bins of the Education attribute were created by enumerating nominal values in the preprocessing stage for data mining. For example, bin "university" was created by merging values "undergraduate" and "graduate" of the underlying "education" data field.

$$\ulcorner \texttt{Education=university} \urcorner$$

The result of transformation to DRL:

```
DrlObj(name == "education", value == "undergraduate"
 || value == "graduate")
```

4.5 Dynamic Binning

A dynamic bin (multiple bins merged into one during mining) is present on attribute *Age* in Rule 2.

$$\ulcorner \texttt{Age=[student} \vee \texttt{senior]} \urcorner$$

The result of transformation to DRL:

```
DrlObj(name == "age", (numVal >= 18 && numVal < 25) ||
 (numVal >= 60 && numVal <= 75))
```

4.6 Conjunction

Conjunction can be featured on the top level within the antecedent or consequent as in Rule 1 and Rule 2, or in a subexpression as in Rule 2.

Top level

$$\ulcorner \text{Amount}{=}\langle 100.000; 200.000) \wedge \texttt{District=Prague} \urcorner$$

This rule fragment is represented in DRL as

```
DrlObj(name == "amount", numVal >= 100000, numVal < 200000)
and DrlObj(name == "district", value == "Praha")
```

Subexpression

$$\ulcorner \texttt{Age=senior} \wedge \texttt{Payments=}\langle 5.000; 10.000) \urcorner$$

This rule fragment is represented in DRL as

```
DrlObj(name == "age", numVal >= 60, numVal <= 75)
and
DrlObj(name == "payments", numVal >= 5000, numVal < 10000)
```

4.7 Disjunction

Disjunction in a simplified GUHA rule can be present only as a subexpression within antecedent or consequent. Disjunction is present in Rule 2:

$$\ulcorner \texttt{(Amount=}\langle 100.000; 200.000) \vee \texttt{Duration=1year)} \urcorner.$$

The result of transformation to DRL:

```
DrlObj(name == "amount", numVal >= 100000, numVal < 200000)
or DrlObj(name == "duration", value == "1year")
```

4.8 Negation

Negation in an EasyMiner rule can be present only on a specific attribute-value pair. Negation is present in Rule 2:

$$\ulcorner \neg \texttt{(district=Bruntal)} \urcorner$$

The result of transformation to DRL:

```
DrlObj(name == "district", value != "Bruntal")
```

This assumes that the value of District is known. An alternative DRL rule, more truthful to the above EasyMiner rule, which would be fired even if the value of District is not available (as in object 2 in Table 1):

```
not DrlObj(name == "district", value == "Bruntal")
```

4.9 Consequent

The specification of the code in rule consequent is currently not yet finalized and the authors would welcome any input from the RuleML community. The provisionary option currently implemented in the system is as follows:

```
then
 processResult(kcontext, "Status", "A");
end
```

The `processResult` is a static method which collects the results of fired rules. Its first argument is a rule context (provided by Drools) followed by the attribute name and its value from consequent of the association rule. As the next step, it is necessary to resolve the situation when multiple rules with different consequents have been activated. One of the options to accomplish this is to use the Drools `accumulate` function.

4.10 Complete DRL

This section lists the complete DRL code for the two example rules.

```
import cz.vse.droolsserver.drools.DrlObj;
import function cz.vse.droolsserver.drools.DrlResult.processResult;

rule "ExampleRule1"
when (
    DrlObj(name == "amount", numVal >= 10000, numVal < 200000)
    and
    DrlObj (name == "district", value == "Prague")
)
then
    // a provisionary construct
    processResult(kcontext, "Status", "A");
end

rule "ExampleRule2"
when (
     (
        DrlObj(name == "amount", numVal >= 100000, numVal < 200000)
        or
        DrlObj(name == "duration", value == "1year")
     )
    and (not DrlObj(name == "district", value == "Bruntal"))
    and
     (
        DrlObj(name == "age", (numVal >= 18 && numVal < 25)
          || (numVal >= 60, numVal <= 75))
        or
        DrlObj(name == "payments", numVal >= 5000, numVal < 10000)
     )
)
then
    // a provisionary construct
    processResult(kcontext, "Status", "B");
end
```

5 Demo Scenario

The demo, accessible at `http://easyminer.eu/demo/ruleml2013`, shows the EasyMiner workflow supporting the business rules integration. All the data-mining steps described in Section 2 are shown as a screencast and as a live demo system. The demo finishes with the user clicking on the *Export as Business Rules* link, which shows the result of converting the rules in the rule clipboard to DRL.

6 Conclusion and Future Work

This paper presents a proof-of-concept system for learning business rules with EasyMiner from transactional data. The main focus of this paper is the transformation of GUHA association rules to the DRL format, used by the open source Drools business rules engine. In this preliminary work we have imposed some restrictions on the form of the GUHA rules being transformed. Nevertheless, the specification proposed here should support all features of conventional association-rule learning algorithms, i.e. those with output similar to the apriori [1] algorithm, plus some advanced features such as disjunctions or negations.

As a future work, we would like to investigate the possibilities for using and extending the human readable serializations of business rules, SBVR "Structured English" [8] in particular, as an alternative way of presenting the discovered rules to the user.

Acknowledgements The work described here was supported by grant IGA 20/2013 of the University of Economics, Prague and by the LinkedTV EU FP7 project.

References

1. Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216. ACM Press, 1993.
2. Bart Goethals and Jan Van Den Bussche. On supporting interactive association rule mining. In *Proceedings of the 2 nd International Conference on Data Warehousing and Knowledge Discovery*, pages 307–316. Springer, 2000.
3. Petr Hájek and Tomáš Havránek. *Mechanizing Hypothesis Formation*. Springer-Verlag, 1978.
4. Tomáš Kliegr, David Chudán, Andrej Hazucha, and Jan Rauch. SEWEBAR-CMS: A system for postprocessing data mining models. In Monica Palmirani, M. Omair Shafiq, Enrico Francesconi, and Fabio Vitali, editors, *RuleML-2010 Challenge*, volume 639 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.
5. Tomáš Kliegr, Andrej Hazucha, and Tomáš Marek. Instant feedback on discovered association rules with PMML-based query-by-example. In *Web Reasoning and Rule Systems*. Springer, 2011.
6. Tomáš Kliegr and Jan Rauch. An XML format for association rule models based on guha method. In *RuleML-2010, 4th International Web Rule Symposium*, Berlin, Heidelberg, 2010. Springer-Verlag.
7. Bing Liu, Yiming Ma, Ching Kian Wong, and Philip S. Yu. Scoring the data using association rules. *Applied Intelligence*, 18(2):119–135, March 2003.
8. OMG (Object Management Group). Semantics of Business Vocabulary and Business Rules (SBVR), v1.0, 2008.
9. Jan Rauch. *Observational Calculi and Association Rules*. Studies in Computational Intelligence. Springer-Verlag, Berlin, 2013.
10. Jan Rauch and Milan Šimůnek. An alternative approach to mining association rules. *Foundation of Data Mining and Knowl. Discovery*, 6:211–231, 2005.

11. Milan Šimůnek and Teppo Tammisto. Distributed data-mining in the LISp-Miner system using Techila grid. In *Networked Digital Technologies'10*, pages 15–21, Berlin, 2010. Springer.
12. Radek Škrabal, Milan Šimůnek, Stanislav Vojíř, Andrej Hazucha, Tomáš Marek, David Chudán, and Tomáš Kliegr. Association rule mining following the web search paradigm. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *ECML/PKDD (2)*, volume 7524 of *Lecture Notes in Computer Science*, pages 808–811. Springer, 2012.