

Finding Maximal Non-Redundant Association Rules in Tennis Data

Daniel Weidner¹, Martin Atzmueller², and Dietmar Seipel¹

¹ University of Würzburg, Department of Computer Science,
Am Hubland, 97074 Würzburg, Germany
{daniel.weidner,dietmar.seipel}@uni-wuerzburg.de

² Tilburg University, Department of Cognitive Science & Artificial Intelligence,
Warandelaan 2, 5037 Ab, Tilburg, The Netherlands
m.atzmueller@uvt.nl

Abstract. The concept of association rules is well-known in data mining. But often redundancy and subsumption are not considered, and standard approaches produce thousands or even millions of resulting association rules. Without further information or post-mining approaches, this huge number of rules is typically useless for the domain specialist – which is an instance of the infamous pattern explosion problem.

In this work, we present a new definition of redundancy and subsumption based on the confidence and the support of the rules and propose post-mining to prune a set of association rules.

In a case study, we apply our method to association rules mined from spatio-temporal data. The data represent the trajectories of the ball in tennis matches – more precisely, the points/times the tennis ball hits the ground. The goal is to analyze the strategies of the players and to try to improve their performance by looking at the resulting association rules. The proposed approach is general, and can also be applied to other spatio-temporal data with a similar structure.

Keywords: Association Rule Mining · Pattern Mining · Post-Mining · Declarative Data Mining · Prolog · Spatio-Temporal Data

1 Introduction

The field of artificial intelligence (AI) can be divided into symbolic and sub-symbolic approaches, e.g., [7, 20, 28, 33]. *Symbolic, knowledge- or rule-based* AI models central cognitive abilities of humans like *logic*, deduction and planning in computers; mathematically exact operations can be defined. *Subsymbolic* or statistical AI tries to learn a model of a process (e.g., an optimal action of a robot or the classification of sensor data), from the data.

Association rules declaratively and symbolically describe logical relations with probabilities in the form of if-then-rules, thus incorporating aspects from both symbolic and statistical approaches. Then, using declarative specifications, e.g., using domain knowledge, specific (inductive) biases, and post-mining approaches, the learning and mining can be supported [5, 7], and post-mining on

the set of association rules – for improving their interestingness and relevancy – can be conveniently implemented. In general, data mining aims to obtain a set of novel, potentially useful and ultimately interesting patterns from a given (large) data set [16]. Here, one prominent method is association rule mining. However, many standard approaches for mining association rules – like the well-known Apriori algorithm – do not consider redundancy or subsumption.

In this paper, we tackle this problem, and demonstrate its application in the spatio-temporal domain of tennis data. Our contributions are summarized as follows: We introduce a new definition of redundant and subsumed association rules to prune the set of rules we obtain from the Apriori algorithm. Furthermore, we present a general post-mining approach for finding maximal non-redundant association rules. Based on the results of previous mining steps, unimportant attributes are excluded in further steps.



Fig. 1. Camera Shot of a Tennis Match.

From an application perspective, analyzing real-world tennis data and pruning the result effectively can lead to individual training methods for the observed players. Furthermore, using data from a video in real-time, a coach could change the player's strategy during a tennis match. Some pre-research had been made in the diploma/master theses [8, 35] and the technical report [31]; by getting information directly from media-data like video-sequences (a screenshot can be seen in Figure 1), essentially the complete (tennis) data mining process can now be automated.

The rest of this paper is structured as follows: Section 2 discusses related work. After that, Section 3 presents the background. Next, we present the proposed data mining process including the new definition of subsumption and

maximality in Section 4. The case study about tennis data shows its usefulness in Section 5. Finally, Section 6 concludes with a summary and lists directions for future work.

2 Related Work

This section discusses related work on association rule mining, condensed representations, and finally post-mining approaches on sets of association rules.

2.1 Association Rule Mining

Association rule mining [1, 2] has been established as a prominent approach in data mining and knowledge discovery in databases, cf. [23] for a survey. Several efficient algorithms have been proposed, including, e.g., the Apriori [2] and the FP-Growth [21] algorithms.

In the research on association rules and finding the most relevant ones, many approaches have been discussed. In [24], the authors present a method to extract rules on user-defined templates or time constraints. In [22], association rules are ranked. This happens with different interestingness measures. Different formats or properties of association rules are discussed in [12]. Finally, the pruning of redundant rules is presented in [14]. Constraint-based data mining also tackles the problem of redundancy in association rule mining. For example, the approach presented in [9] presents an approach for constraint-based rule mining in large, dense databases, focusing on the interestingness of specializations of rules relative to their parent generalization with specific thresholds.

In contrast to the approaches discussed above, we employ a standard method for association rule mining (e.g., Apriori) which is customized using logic programming, such that the mining step can be re-iterated in a declarative way.

2.2 Condensed Representations and Post-Mining Approaches

Condensed representations of association rules for reducing redundancy mainly focus on closed itemsets, e.g., [6, 39]. Furthermore, also research in the domain of formal concept analysis has resulted in several algorithms, e.g., [29, 34]; also cf. [10] for a survey on condensed representations. Furthermore, [17] presents a mining approach for finding the top- k non-redundant association rules using an approximation algorithm.

Considering post-mining methods, [40] discusses several techniques for effective knowledge extraction from association rules, while [25–27] apply ontologies to facilitate the post-processing of a set of association rules, also including interaction with a domain expert. Logic-based post-mining approaches include a technique where patterns are filtered using constraints formulated with answer set programming (ASP) [19].

Similar to the approaches described above, we also apply post-mining but using declarative techniques. We apply formalizations of subsumption and redundancy for declaratively shaping the association rules. However, specifically

in contrast to the existing logic-based approaches, the presented approach is not restricted to work on the set of association rules directly, but can further refine the mechanism of how to discover association rules, by e. g., refining the data representation, the parameters of the mining process, and its subsequent results at the same time in incremental fashion.

3 Association Rules and the Apriori Algorithm

In this section, we provide an overview on the relevant background on association rules, before briefly summarizing the Apriori algorithm.

3.1 Association Rules

We consider a set of transactions, where each transaction is a set of items, called an itemset. An *association rule* $r = L \Rightarrow R$ is a classification rule, where the antecedent L and the conclusion R are itemsets; without loss of generality, we assume $L \cap R = \emptyset$. The *support* of an association rule is the number of the transactions containing both sides divided by the number of all transactions. The *confidence* of an association rule expresses the likelihood that R occurs in a transaction, if L occurs in the transaction. It is defined as the percentage of transactions containing $L \cup R$ among the transactions containing L . We write $sup(r)$ and $conf(r)$ for the support and confidence, respectively. Note, that support and confidence do not depend on each other, and both definitions are necessary for association rule mining. There can be rules with a large support but a small confidence, and vice versa.

The main goal of association rules mining is to find rules having a minimum confidence and support. These rules may be obvious for an expert, but we will show in a case study for tennis data that they can reveal new, unknown relations.

3.2 The Apriori Algorithm

The Apriori algorithm is a standard method for finding association rules according to specified minimal support and minimal confidence thresholds. This algorithm incrementally searches for frequent itemsets, utilizing the minimal support threshold. It starts with itemsets of size 1, and iteratively refines the itemsets, enlarging the itemsets. The algorithm stops, if there is no frequent itemset of a certain size. From all frequent k -itemsets, all possible $(k + 1)$ -itemsets are considered if they are also frequent, i. e., they exceed the minimum support.

In the context of this paper, we apply a specific implementation of the Apriori algorithm, provided by the well-known data mining tool Weka [37]. In Section 5.2, we will see, that Weka can be used with a parameter which stands for a required number of rules. So in the Apriori algorithm implemented in Weka, the support is reduced until this number of rules is reached. Depending on the size and characteristics of the data set, this can potentially be a very large number of rules, that also exceeds a given minimal confidence. Depending on the size

of the table and the minimal confidence, this number of rules can be unusably high, so that users lose track of those rules. At this point, the process has to be run again with other parameters or some expert needs to filter important rules, which ends in looking for a needle in a haystack. Since both of these methods are very time-consuming and expensive, we propose a new approach by pruning rules effectively. For this, we employ the idea of redundancy and subsumption, which we define in the next section in detail.

4 Data Mining Process

The proposed declarative data mining process, which will be discussed in this section, is implemented within the software package *Declare* [15] for knowledge-based intelligent systems that is developed using *SWI-Prolog* [36]. We introduce a (semi-) automatic data mining process that consists of the steps outlined below. By repeating these steps with different parameters or transformations, we achieve different results in each iteration, until some result is convincing enough according to the assessment of a domain expert. Thus, the evaluation of the results has to be supported by a domain expert or some knowledge structure like a knowledge graph. In the latter case, the process can then also be potentially automatized, which we plan to do in the future; so far the process workflow is semi-automatic. The process workflow is presented in Figure 2. In the following, we will present the steps of an iteration.

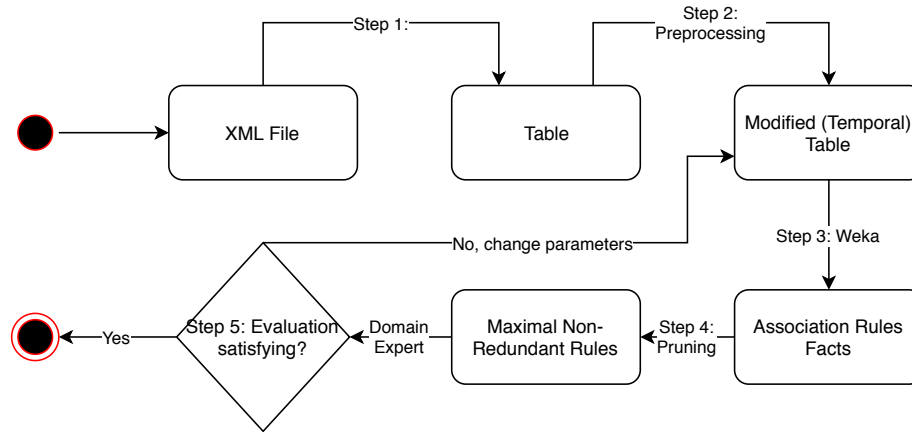


Fig. 2. Flowchart of the Data Mining Process.

Step 1 and 2: From XML to Modified (Spatio–Temporal) Table (Feature Selection and Extraction)

Given an XML file with data to be analyzed, we want to create a structured table. For this, some aspects should be discussed. First of all, we have to think about which attributes of the XML file should be included in the table and the data mining process. In our test data we skip some unimportant attributes by projecting the table. By now, the selected attributes are chosen first, then the algorithm is run. In the future we plan to make this interactive, so the user is asked, which attributes should be involved in one iteration. As it can be seen in Section 5.2, we did not just use the attributes of the XML file, we also transform attributes, which are exact coordinates, so the size of the domain equals the size of the table, and the attributes appear with a probability near to zero. Another transformation is done to create a temporal data scheme. Both transformations may be applicable to other XML files, but it is impossible to guide this automatically. Nevertheless, if users are familiar with the used file, they should think about such transformations, since attributes with a probability near to zero will only reach a small support and confidence, and a temporal data table allows other data mining methods too, see [3].

Step 3: From a Table to Association Rules (via Weka)

As discussed in Section 3.2, we use the Apriori algorithm of Weka to get association rules. This step includes two thresholds that can be modified in each iteration: first the number of required rules can be increased, and second the value of the minimum confidence can be decreased. This may lead to a greater number of rules: so the bigger the search space of rules, the higher is the chance to find very interesting rules.

Step 4: Pruning Non–Maximal Redundant Association Rules

After all found association rules are loaded into the system, we want to get rid of redundant and non–maximal association rules. For this, we need a new definition of redundancy, where an association rule $r_1 = L_1 \Rightarrow R_1$ is called *redundant*, if there is another association rule $r_2 = L_2 \Rightarrow R_2$, such that

$$L_2 \subseteq L_1, R_1 \subseteq R_2 \text{ and } \text{conf}(r_2) = 1.$$

Note that, if a rule r_1 is redundant, then its confidence does not have to be 1 in general; e.g., for a redundant rule $r_1 = L_1 \Rightarrow R_1$ and a rule $r_2 = L_2 \Rightarrow R_2$ with $\text{conf}(r_2) = 1$, such that $L_2 = L_1$ and $R_1 \subsetneq R_2$, we have $\emptyset = L_2 \cap R_2 = L_1 \cap R_2$, and we get $1 = \text{conf}(r_2) = \frac{|L_2 \cup R_2|}{|L_2|} = \frac{|L_1 \cup R_2|}{|L_1|} > \frac{|L_1 \cup R_1|}{|L_1|} = \text{conf}(r_1)$. Our definition of redundancy is different from related literature like [17, 18, 24].

We say that an association rule $r_1 = L_1 \Rightarrow R_1$ subsumes another association rule $r_2 = L_2 \Rightarrow R_2$, in short $r_1 \supseteq r_2$, if

$$L_1 \subseteq L_2, R_2 \subseteq R_1 \text{ and } \text{sup}(r_1) \geq \text{sup}(r_2), \text{ conf}(r_1) \geq \text{conf}(r_2).$$

A rule r is called *maximal*, if it is not subsumed by any other rule $r' \neq r$. Both definitions have first appeared in the lecture [32]; like support and confidence they do not depend on each other. This means, a subsumed rule can be non-redundant, and a redundant rule may not be subsumed by any other rule. After applying these definitions, we hope to finally obtain a small number of maximal non-redundant rules.

In the following, we motivate these definitions. Essentially, to find interesting association rules, these definitions are necessary. This is also shown by Table 3, where we ran Weka with the same table as in Section 5, but searching for all possible association rules. This means the minimum confidence is increased by 0.1, and we count the number of all association rules found by Weka. For an example with tennis data, which will be presented in detail in Section 5, we found about 22 000 rules, even if we required a minimum support of 1. And with a standard minimum confidence of 0.50 or 0.75, we reach 70 000 or 40 000 rules. As said before, searching for the set containing the most interesting rules of this large total rule-set is nearly impossible; but it is reasonable to work with a small number of maximal non-redundant rules to obtain unknown relations.

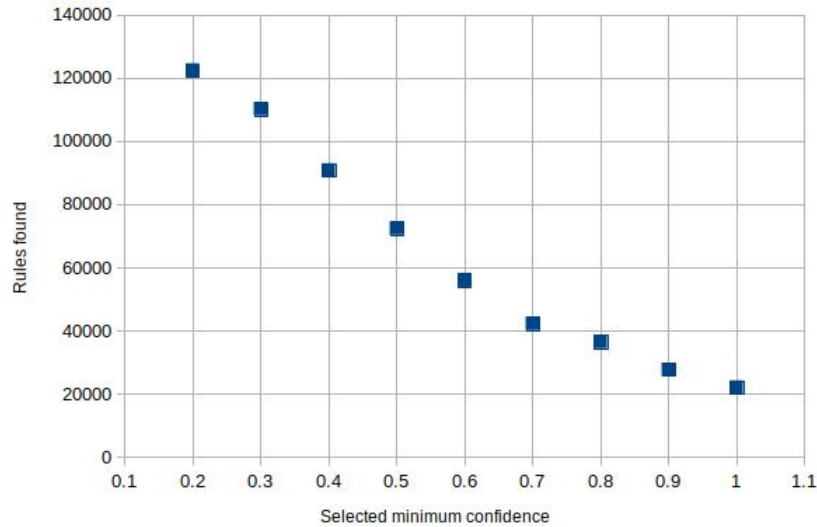


Fig. 3. Relation between Minimum Confidence and Number of Found Association Rules in the Tested Tennis Data Set.

Step 5: Iterating the Process with New Parameters

After one iteration, some evaluation of the result has to be done. Depending on the quality of the evaluation, the system should repeat the process with different

parameters, starting with those for Weka. This is because these thresholds need no information about the initial data. It can be increased and decreased, respectively, to create more association rules. If all these iterations fail the evaluation, then some modification of the starting table and attributes should be done. For this, the user needs information or knowledge about the data to guide the data mining process, i. e., in- and excluding the right attributes and changing the right parameters correctly.

5 Case Study: Analysis of Tennis Data

A system for the management and analysis of tennis data had been started in SWI-Prolog in the diploma thesis [35], where an XML representation had been developed (see Listing 1 below), and some simple analysis had been done. Later, this analysis had been extended with a functionality to query the data [31] using the Prolog-based XML processing utilities of [30]. Prolog is very useful here, since knowledge bases with semi-structured, symbolic data, such as relational, deductive, XML or semantic web data can be handled nicely with Prolog [11, 13].

In the following subsections, we are *refining* the proposed *data mining process* for deriving suitable association rules. First, the XML file with the tennis data is transformed into a relational table. Columns are created from the attributes of the file; attributes can be omitted, if they should not be involved in the mining process. This initial table is transformed into a modified, temporal table; some of these modifications are not universally applicable, but key ideas may be portable to other types of data. First, we create a *tessellation* for the tennis court, since exact coordinates will be repeated with a probability near to zero. Second, we duplicate a part of the table in order to model the data in a special way, such that traditional data mining is lifted to *temporal* data mining.

Then, the association rules are computed using the Apriori algorithm for frequent itemsets of the tool Weka. After wrapping the Weka output text file of the rules into Prolog facts and consulting them, the *maximal non-redundant rules* are filtered as the desired association rules.

5.1 Preparing the Data: Creating and Duplicating the Table

We start with a given XML file in the format of [35]; an example is given in Listing 1. Here the main information is saved in the (sub-) elements `set`, `game`, `point` and `hit`. These attributes will mostly form the columns of our table. For a different file, a similar approach is conceivable.

Listing 1. XML File of a Tennis Match.

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<match>
  <player id="A" name="Sampras"/>
  <player id="B" name="Agassi"/>
  <result>
```



```

    <score set="1" player_A="6" player_B="3"/> ...
  </result>
  <match_facts>
    <tournament>US Open 2002</tournament> ...
  </match_facts>
  <set id="1" score_A="5" score_B="3">
    <game id="1" service="A" score_A="0" score_B="0">
      <point id="1" top="B" service="A" score_A="0"
        score_B="0" winner="A" error="0">
        <hit id="1" hand="forehand" type="ground"
          time="00:00:42" x="0.17" y="-12.07"/>
        <hit id="2" hand="backhand" type="ground"
          time="00:00:44" x="-0.49" y="5.89"/>
        <hit id="3" hand="backhand" type="ground"
          time="00:00:46" x="-3.92" y="-3.42"/>
        <hit id="4" hand="forehand" type="ground"
          time="00:00:48" x="3.56" y="2.06"/>
      </point> ...
    </game> ...
  </set> ...
</match>

```

This file is transformed into a relational table, where all information is saved, see Figure 4.

hit	set	game	point	top	service	score_A	score_B	winner	error	hand	type	time	x	y
hit1	set:1	game:1	point:1	top:B	service:A	score_A:0	score_B:0	winner:A	error:0	hand:forehand	type:ground	time:00:00:42	x:0.17	y:-12.07
hit2	set:1	game:1	point:1	top:B	service:A	score_A:0	score_B:0	winner:A	error:0	hand:backhand	type:ground	time:00:00:44	x:-0.49	y:5.89
hit3	set:1	game:1	point:1	top:B	service:A	score_A:0	score_B:0	winner:A	error:0	hand:backhand	type:ground	time:00:00:46	x:-3.92	y:-3.42
hit4	set:1	game:1	point:1	top:B	service:A	score_A:0	score_B:0	winner:A	error:0	hand:forehand	type:ground	time:00:00:48	x:3.56	y:2.06
hit1	set:1	game:1	point:2	top:B	service:A	score_A:1	score_B:0	winner:A	error:0	hand:forehand	type:ground	time:00:01:19	x:-0.27	y:-12.03
hit2	set:1	game:1	point:2	top:B	service:A	score_A:1	score_B:0	winner:A	error:0	hand:forehand	type:ground	time:00:01:21	x:0.08	y:8.06
hit1	set:1	game:1	point:3	top:B	service:A	score_A:2	score_B:0	winner:A	error:0	hand:forehand	type:ground	time:00:01:42	x:0.51	y:-12.03
hit2	set:1	game:1	point:3	top:B	service:A	score_A:2	score_B:0	winner:A	error:0	hand:forehand	type:ground	time:00:01:45	x:-3.92	y:2.97

Fig. 4. Part of the Initial Relational Table for the Tennis Data.

At this point, we do some small modification, which is exclusive for the tennis data. If we save the exact coordinates where the ball hits the ground, then the same spot will be repeated with a probability near to zero. So we create a *tessellation* for the court in $N \times M$ regions and instead of the exact x - and y -coordinates we save which intervals are reached, for example the red box 1 inside the court. Note that also the outside of the court forms intervals with numbers 1 and $N + 2$ or $M + 2$, respectively, (e. g., the blue box number 2), see Figure 5.

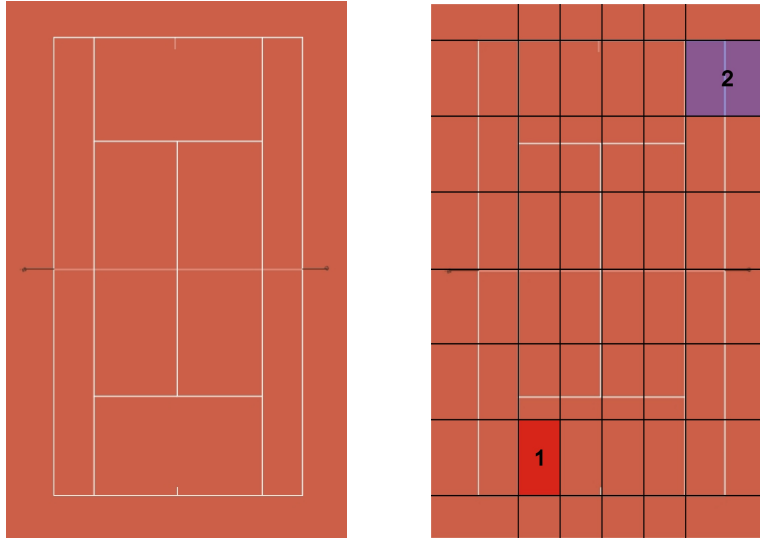


Fig. 5. Tessellation of a Tennis Court.

Next, we create the temporal table, by duplicating part of the table of Figure 4. Here, we save the information for a hit and the next hit, if there is one for this point. This means, that if a point has a hit order 1-2-3, we have the rows of hit pairs (1,2) and (2,3). In the resulting rows, all attributes are saved. But, since not all information is useful, we omit some which might not be interesting for the data mining process; from Listing 1, we skip `set`, `game`, `top`, `service`, `score_A`, `score_B`; also attributes for the temporal data, namely `type_{1,2}`, `time_{1,2}`, `x_{1,2}`, `y_{1,2}`, are skipped. Thus, we obtain the table of Figure 6. Here the red boxes show the temporal data. Note that the first attribute `hit` stands for hit pairs; for example in the fourth point with the hits 1-2-3-4-5, we get the four hit pairs (1,2), (2,3), (3,4) and (4,5) (see the blue boxes).

This transformation could be modified, such that we save hit triples. Also different numbers of `x`- and `y`- intervals lead to different association rules. The modification of the table may be supported by a domain expert. In future work, we plan to involve some knowledge at this point.

5.2 From Table to Association Rules

The input file `weka_input.arff` of the data mining tool Weka describes the attributes, their domain and the rows of the table. The other additional parameters `-N 4000` and `-C 0.5` in the Weka call given in Listing 2 stand for the required number of association rules (here 4000) and the minimum confidence (here 0.5), respectively; their default values would be 10 and 0.9. Then we get an output text file `weka_output`.

hit	point	winner	error	tx_1	ly_1	tx_2	ly_2	id_1	hand_1	id_2	hand_2
hit1	point:1	winner:A	error:0	tx_1:3	ly_1:0	tx_2:2	ly_2:6	id_1:1	hand_1:forehand	id_2:2	hand_2:backhand
hit2	point:1	winner:A	error:0	tx_1:2	ly_1:6	tx_2:1	ly_2:2	id_1:2	hand_1:backhand	id_2:3	hand_2:backhand
hit3	point:1	winner:A	error:0	tx_1:1	ly_1:2	tx_2:4	ly_2:5	id_1:3	hand_1:backhand	id_2:4	hand_2:forehand
hit1	point:2	winner:A	error:0	tx_1:2	ly_1:0	tx_2:3	ly_2:7	id_1:1	hand_1:forehand	id_2:2	hand_2:forehand
hit1	point:3	winner:A	error:0	tx_1:3	ly_1:0	tx_2:1	ly_2:5	id_1:1	hand_1:forehand	id_2:2	hand_2:backhand
hit1	point:4	winner:A	error:0	tx_1:2	ly_1:0	tx_2:4	ly_2:6	id_1:1	hand_1:forehand	id_2:2	hand_2:backhand
hit2	point:4	winner:A	error:0	tx_1:4	ly_1:6	tx_2:2	ly_2:1	id_1:2	hand_1:backhand	id_2:3	hand_2:forehand
hit3	point:4	winner:A	error:0	tx_1:2	ly_1:1	tx_2:1	ly_2:7	id_1:3	hand_1:forehand	id_2:4	hand_2:forehand
hit4	point:4	winner:A	error:0	tx_1:1	ly_1:7	tx_2:2	ly_2:4	id_1:4	hand_1:forehand	id_2:5	hand_2:forehand
hit1	point:1	winner:B	error:0	tx_1:3	ly_1:0	tx_2:1	ly_2:6	id_1:1	hand_1:forehand	id_2:2	hand_2:forehand

Fig. 6. Part of the Temporal Tennis Data.

Listing 2. Bash Command to Call Weka.

```
java -cp ./weka.jar weka.associations.Apriori -t
data/weka_input.arff -N 4000 -C 0.5 > data/weka_output
```

We call this bash command inside Prolog via the built-in predicate `unix/1` to create the output, that looks like Listing 3. Weka sorts the rules by the confidence and stops once the number of required rules is reached.

Listing 3. Fragment of the Weka Output File.

```
Apriori
=====
Minimum support: 0.15 (409 instances)
Minimum metric <confidence>: 0.5
Number of cycles performed: 17
Generated sets of large itemsets:
Size of set of large itemsets L(1): 40 ...
Size of set of large itemsets L(9): 1

Best rules found:
1. service=B hand_1=forehand 1077 ==>
   type_2=ground 1077 conf:(1)
2. service=B hand_1=forehand type_1=ground 1072 ==>
   type_2=ground 1072 conf:(1) ...
```

5.3 From Association Rules to Facts

From the output file obtained by Weka, we create corresponding Prolog facts for the found association rules. Since an association rule has four characteristic attributes, namely antecedent, consequent, support and the confidence, we save them in addition to a unique identifier. In particular we get facts of the form

```
rule(Id, Ant, Cons, Sup, Conf),
```

where **Ant** and **Cons** are lists that represent the itemsets. These facts are obtained using the Prolog-based parsing and XML processing utilities of **Declare**. After loading them into the system, the user can query them in Prolog. Currently, we are also experimenting with other programming languages like Python for working with strings and text files.

5.4 From Facts to Maximal Non-Redundant Rules

From the facts for the association rules, we compute the redundancy and subsumption in **Declare**. Listing 4 defines in Prolog when the rule with the identifier **Id_1** is redundant because of the rule **Id_2**, and when the rule with **Id_1** is subsumed by the rule **Id_2**:

Listing 4. Definition of Redundant and Subsumed Rules in **Declare**.

```

redundant_rule(Id_1, Id_2) :-
    rule(Id1, Ant1, Cons1,_,_),
    rule(Id2, Ant2, Cons2,_,1),
    Id1 =\= Id2,
    subset(Ant2, Ant1), subset(Cons1, Cons2).

subsumed_rule(Id_1, Id_2) :-
    rule(Id_1, Ant1, Cons1, Sup1, Conf1),
    rule(Id_2, Ant2, Cons2, Sup2, Conf2),
    Id_1 =\= Id_2, Sup2 > Sup1, Conf2 > Conf1,
    subset(Ant2, Ant1), subset(Cons1, Cons2).

```

For our example with the 4×6 -tessellation, these Prolog rules have derived 46 maximal non-redundant rules, 3 of which are given in Listing 5.

Listing 5. Example of Maximal Non-Redundant Rules.

```

41:  [id_2=2] => [hit=1,hand_1=forehand,id_1=1]
      Sup:0.31 Conf:1
42:  [id_1=1] => [hit=1,hand_1=forehand,id_2=2]
      Sup:0.31 Conf:1
416: [Iy_2=5] => [hand_1=forehand] Sup:0.18 Conf:0.88

```

It is now an advanced task to find the rules, which are useful for trainers or players. Here, domain knowledge from experts has to be included.

5.5 Experimental Results

In the following, some experiments and their results are discussed. We compare the number of maximal non-redundant rules with the maximal number of rules and the minimum confidence in the Apriori algorithm of Weka. Table 1 shows that the 4×6 -tessellation leads to a small (and so manageable) number of rules

together with our definitions of redundancy and subsumption. In most cases, less than 1% of the required rules are maximally non-redundant. Only if we choose a minimum confidence of 0.5, then we get up to 5%. Nevertheless, the total number of rules is useful, and a domain expert has to check only a manageable number of interesting rules. Thus, the new definitions of subsumption and maximality lead to convincing results for the tennis data.

Required Rules	Minimum Confidence	Maximal Non-Redundant Rules
5000	0.5	46
7500	0.5	28
10000	0.5	156
10000	1.0	38
12500	0.5	442
12500	1.0	42
15000	0.5	773
15000	0.75	48

Table 1. Experimental Results.

We have manually tested many different parameters for the tessallations. In the future, we would like to automatize the process of searching for suitable parameters for convincing association rules, which can be used as initial parameters in further experiments. This will increase our experience about parameters leading to good results.

6 Conclusions and Future work

In this paper, we have introduced new definitions of redundant and subsumed association rules, respectively. With these definitions, we have discussed a data mining process. In a case study, the definitions have proven useful and led to results, which are far more convincing than the results of the standard Apriori algorithm.

In the future, we are considering to use Datalog and answer set programming for guiding an automatized data mining process. Again, the handling of the symbolic data (relations, deduction or association rules) will be done in logic programming. In particular, we are planning to automatize parts of the data mining workflow given in Figure 2 to decide on suitable parameters for the next iteration based on an analysis of the previously derived association rules. We will also consider other pattern mining approaches, e. g., subgroup discovery [4, 38].

References

1. Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proc. ACM SIGMOD Conf. Washington D.C.*, volume 22, pages 207–216. ACM, 1993.

2. Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of the 20th VLDB Conference, Santiago Chile*, pages 487–499, 1994.
3. Cláudia M. Antunes and Arlindo L. Oliveira. Temporal Data Mining: an overview. In *KDD workshop on temporal data mining*, volume 1, pages 1–13, 2001.
4. Martin Atzmueller. Subgroup Discovery. *WIREs DMKD*, 5(1):35–49, 2015.
5. Martin Atzmueller and Dietmar Seipel. Declarative Specification of Ontological Domain Knowledge for Descriptive Data Mining. In *Proc. International Conference on Applications of Declarative Programming and Knowledge Management (INAP)*, pages 158–170, 2007.
6. Yves Bastide, Nicolas Pasquier, Rafik Taouil, Gerd Stumme, and Lotfi Lakhal. Mining Minimal Non-Redundant Association Rules Using Frequent Closed Itemsets. In *CL 2000 international conference on Computational Logic*, pages 972–986. Springer, 2000.
7. Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational Inductive Biases, Deep Learning, and Graph Networks. *arXiv preprint arXiv:1806.01261*, 2018.
8. Michael Baumgart. Erkennung von Spielstand, Schlagposition und Spielertrajektorien beim Tennis. Master Thesis, University of Würzburg, 2019.
9. Roberto J. Bayardo, Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-Based Rule Mining in Large, Dense Databases. *Data Mining and Knowledge Discovery*, 4(2-3):217–240, 2000.
10. Boulicaut, Jean-François. *Encyclopedia of Data Warehousing and Mining*, chapter Condensed Representations for Data Mining, pages 207–211. Idea Group, 2005.
11. Ivan Bratko. *Prolog Programming for Artificial Intelligence*. Addison-Wesley Longman, 4th edition, 2011.
12. Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond Market Baskets: Generalizing Association Rules to Correlations. *Proceedings of the ACM Conference on Management of Data (SIGMOD)*, pages 265–276, 1997.
13. William Clocksin and Christopher S. Mellish. *Programming in Prolog*. Springer Science & Business Media, 2003.
14. Laurentiu Cristofor and Dan Simovici. Generating an informative cover for association rules. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 597–613, 2002.
15. Declare – A Declarative Toolkit for Knowledge-Based Systems and Logic Programming. <http://www1.pub.informatik.uni-wuerzburg.de/databases/research.html>.
16. Fayyad, Usama and Piatetsky-Shapiro, Gregory and Smyth, Padhraic. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37–54, 1996.
17. Philippe Fournier-Viger and Vincent S. Tseng. Mining Top-K Non-Redundant Association Rules. In *International Symposium on Methodologies for Intelligent Systems (ISMIS)*, pages 31–40. Springer, 2012.
18. Philippe Fournier-Viger and Vincent S. Tseng. TNS: Mining Top-K Non-Redundant Sequential Rules. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 164–166. ACM, 2013.
19. Martin Gebser, Thomas Guyet, René Quiniou, Javier Romero, and Torsten Schaub. Knowledge-Based Sequence Mining with ASP. In *5th International joint conference on artificial intelligence (IJCAI)*, pages 8–15, 2016.

20. Ben Goertzel. Perception Processing for General Intelligence: Bridging the Symbolic/Subsymbolic Gap. In *International Conference on Artificial General Intelligence*, pages 79–88. Springer, 2012.
21. Jiawei Han, Jian Pei, and Yiwen Yin. Mining Frequent Patterns Without Candidate Generation. In *In: Proceedings of ACM SIGMOD international conference on management of data*, pages 1–12. ACM Press, 2000.
22. Robert J. Hilderman and Howard J. Hamilton. *Knowledge Discovery and Measures of Interest*, volume 638. Springer Science & Business Media, 1999.
23. Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for Association Rule Mining – A General Survey and Comparison. *SIGKDD Explorations*, 2(1):58–64, 2000.
24. Sotiris Kotsiantis and Dimitris Kanellopoulos. Association Rules Mining: A Recent Overview. *GESTS International Transactions on Computer Science and Engineering*, 32(1):71–82, 2006.
25. Mansingh, Gunjan and Osei-Bryson, Kweku-Muata and Reichgelt, Han. Using Ontologies to Facilitate Post-Processing of Association Rules by Domain Experts. *Information Sciences*, 181(3):419–434, 2011.
26. Claudia Marinica and Fabrice Guillet. Knowledge-Based Interactive Postmining of Association Rules using Ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 22(6):784–797, 2010.
27. Claudia Marinica, Fabrice Guillet, and Henri Briand. Post-Processing of Discovered Association Rules using Ontologies. In *2008 IEEE International Conference on Data Mining Workshops*, pages 126–133. IEEE, 2008.
28. Clayton McMillan, Michael C Mozer, and Paul Smolensky. Rule Induction through Integrated Symbolic and Subsymbolic Processing. In *Advances in Neural Information Processing Systems*, volume 4, pages 969–976, 1992.
29. Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering Frequent Closed Itemsets for Association Rules. In *ICDT&AŁ1999 International Conference on Database Theory*, pages 398–416. Springer, 1999.
30. Dietmar Seipel. Processing XML-Documents in Prolog. In *Workshop on Logic Programming (WLP 2002)*, 2002.
31. Dietmar Seipel. Analyse von Tennismatches am Beispiel des Finales der US Open 2002: Pete Sampras – Andre Agassi, 2004.
32. Dietmar Seipel. *Advanced Databases*, Course at the University of Würzburg, 2018.
33. Paul Smolensky. Connectionist AI, Symbolic AI, and the Brain. *Artificial Intelligence Review*, 1(2):95–109, 1987.
34. Gerd. Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing Iceberg Concept Lattices with Titanic. *Data and Knowledge Engineering, Elsevier*, 42(2):189–222, 2002.
35. Jürgen Wehner. Verwaltung und Analyse von Zeitreihen zu Videosequenzen. Diploma Thesis, University of Würzburg, 2003.
36. Jan Wielemaker. SWI-Prolog Reference Manual 7.6. Technical report, 2017.
37. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, 2000.
38. Stefan Wrobel. An Algorithm for Multi-Relational Discovery of Subgroups. In *Proceeding of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 78–87. Springer, 1997.
39. Mohammed J. Zaki. Generating Non-Redundant Association Rules. In *6th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*, pages 34–43, 2000.
40. Yanchang Zhao. *Post-Mining of Association Rules: Techniques for Effective Knowledge Extraction*. Hershey, 2009.