Prague University of Economics and Business
Faculty of Informatics and Statistics

Experiment with rule mining from linked government data

MASTER THESIS

Study program: Applied Informatics
Field of study: Knowledge and Web Technologies

Author: Bc. Petr Novák
Supervisor: prof. Ing. Vojtěch Svátek, Dr.

Prague, June 2021

Acknowledgements

Thank you to prof. Ing. Vojtěch Svátek, Dr. for the factual advice and professional guidance of this work.

Abstrakt

Některé organizace veřejného sektoru a vládní orgány zveřejňují svá data jako LOD datové kostky. Vzájemně propojená povaha LOD vybízí k tomu, aby byly tyto datové kostky obohaceny o další informace dostupné z jiných zdrojů publikovaných také jako RDF. Tyto nové informace ve formě binárních vztahů obsažené v těchto znalostních grafech lze použít při těžbě asociačních pravidel nad těmito kostkami, což může vést k nalezení vztahů, které nelze najít v kostkách samotných. Těžba asociačních pravidel nad daty RDF a zároveň v jejich agregované podobě je dosud neprozkoumanou oblastí a nalezení smysluplných interpretovatelných pravidel, která přinášejí nové poznatky, není dosud vyřešeným problémem. Tato práce zkoumá možnosti obohacení RDF datové kostky strukturované slovníkem Data Cube Vocabulary o data ze znalostních grafů a těžby těchto dat pomocí algoritmu AMIE a jeho jiných variant. Zjištění jsou demonstrována na provedeném experimentu těžby asociačních pravidel s frameworkem RDFRules na datasetech České správy sociálního zabezpečení, Českého statistického úřadu, Wikidata a YAGO.

Klíčová slova

Data mining, OLAP, RDFRules, AMIE, Propojená data, RDF, Asociační pravidla

Abstract

Some public-sector organizations and governmental bodies are publishing their data as LOD data cubes. The interlinked nature of LOD encourages the published data cubes to be enriched with additional information available from other sources published as RDF as well. This new information in the form of binary relationships contained in these knowledge graphs can be used when mining association rules over the aggregated data, which can lead to finding relationships that cannot be found in the cubes themselves. Mining of association rules over RDF data and at the same time in their aggregated form is a not yet explored area and achieving the generation of meaningful interpretable rules that bring new knowledge is not yet a solved problem. This work explores the possibilities of enriching the RDF data cube structured by the Data Cube vocabulary with the data from general knowledge graphs and of mining such data by the AMIE algorithm or its derivatives. The findings are demonstrated in a performed experiment of mining association rules with the RDFRules framework over the data sets of the Czech Social Security Administration, Czech statistical office, Wikidata, and YAGO.

Keywords

Data Mining, OLAP, RDFRules, AMIE, Linked Data, RDF, Association Rules

Contents

Introduction

OLAP [5] or data cubes are established as one of the standard tools for analyzing aggregated data and are an essential part of any Bussiness Intelligence solution. Any transactional data, that relates to multiple contexts (e.g. multinational fast-food chain's sales can be divided with respect to regions, marketing campaigns, etc.) can be transformed into their multidimensional representation in a form of OLAP cube where the contexts potentially relevant for analysis make up the dimensions of the cube and be examined in one of many available standard BI tools (such as Microsoft Excel, Power BI, etc.), that facilitate aggregation of the values and manipulation with the cube in order to gain insights into the data.

Data mining can function as an alternative or complement [12] to OLAP analysis. The term data mining refers to extracting or *mining* knowledge from large amounts of data [10], usually relational databases where the data is on the most granular level. The purpose of data mining is either *descriptive*, where it is aimed to find and describe regularities and patterns in the data, or *predictive*, where the goal is to infer new information hidden in the data, that can be used to make predictions. Association rules are one of the data mining methods and can be used for descriptive and predictive analysis. The rules can be mined by different algorithms, namely the Apriori algorithm [2] and the ASSOC procedure of the GUHA method.[9] They both serve to mine association rules in tabular data.

Data representation is not only limited to tables. In the environment of the World Wide Web, data that is intended to be published to the public is often modeled according to the Resource Description Framework (RDF) model. Individual records in the RDF model take the form of *triples*, that represent binary relationships between the described entities. Sets of the relationships between entities and concepts that those entities can represent are call *vocabularies* or *ontologies* and are published as RDF as well. The relationships can connect entities from different data sources, making the data *linked*. The term *Linked Open Data* (LOD) has become established for RDF data, which adheres to the use of standard formats, technologies, and interconnection principles that facilitate and expand the possibilities of working with this data.

An algorithm, that was designed specifically for mining rules similar to association rules from the RDF data in the Web environment is called ***A**ssociation Rule **M**ining under **I**ncomplete **E**vidence*, shorty AMIE. [8] Since the release of the original version of the algorithm, its original authors have published its two extensions, called AMIE+ [7] and AMIE 3 [15], respectively. Another extension that builds on the AMIE + version, called RDFRules [26], comes with its reference implementation in the form of a robust framework, which focuses not only on the rule mining itself but also on the possible preprocessing of input data and processing the generated rules.

Public-sector organizations and governmental bodies, that manage a large amount of data including various registers and demographic and economic statistics are often incentivized to

publish some of their data for the benefit of their citizens. Due to data privacy restrictions [19], the organizations often resort to publishing their data in an aggregated form. Some of the organizations, such as the European Union through its *OpenBudgets.eu*[1] platform or the Czech Social Security Administration[2], acknowledge the advantages of publishing their data in a universal and standard way for the consumers and do not hesitate to devote their resources to publishing their data as LOD. The Data Cube vocabulary [6] is used to write aggregated data in the form of a data cube in the RDF model.

The interlinked nature of LOD encourages the published data cubes to be enriched with additional information available from other sources published as RDF as well. The connection would be made through those data cube dimension values that simultaneously occur as objects in large cross-domain LOD data sets known as *knowledge graphs*. Those can be municipalities, regions, organizations, products, etc. This new information in the form of binary relationships contained in these knowledge graphs can be used when mining association rules over the aggregated data, which can lead to finding relationships that cannot be found in the cubes themselves. Mining of association rules over RDF data and at the same time in their aggregated form is a not yet explored area and achieving the generation of meaningful interpretable rules that bring new knowledge is a not yet solved problem.

The goals of this work are to:

1. explore the possibilities of enriching RDF data cube structured by the Data Cube vocabulary with the data from general knowledge graphs and of mining such data by the AMIE algorithm or its derivatives,
2. carry out an experiment that would demonstrate that such an approach is possible and capable of yielding new interesting insights on the aggregated data.

The RDFRules framework is one of the tools used for performing the experiment. The aggregated data examined in the experiment are Czech pension statistics published by Czech Social Security Administration and Czech demography statistics of the Czech statistical office[3]. The triples associated with the dimension values of the data cube are extracted from Wikidata[4] and YAGO[5] knowledge graphs.

This paper is organized as follows. Section 1 provides an overview of Linked Open Data principles, the RDF data model and its serialization formats, and the data sets used in the experiment. Section 2 introduces the basic concepts of OLAP cubes and describes the vocabularies used for representing the data cubes in RDF. Section 3 explains the basics of the association rules. Section 4 describes the AMIE algorithm, the improvements of its extension AMIE+ and presents the RDFRules algorithm and the techniques suggested to be used along with it. Section 5 describes the reference implementation of the RDFRules.

[1] https://ec.europa.eu/digital-single-market/en/content/openbudgetseu
[2] https://data.cssz.cz/
[3] https://www.czso.cz/csu/czso/home
[4] https://www.wikidata.org/
[5] https://yago-knowledge.org/

Section 6 elaborates the possibilities and implications of mining rules over RDF data cubes with a combination of triples from knowledge graphs. Section 7 describes thoroughly the performed experiment. Section 8 elaborates the results of the executed mining tasks and encountered problems and the conclusion summarizes the contribution of the work and gives suggestions for further activities.

1. Linked Data

Linked Data is a set of best practices for publishing and connecting data on the Web structured in such a way that it is usable not only for human processing but also processable by machines. It builds upon the general architecture of the World Wide Web. Instead of creating links between particular documents from different sources in the case of the classic Web of documents, the Linked Data connects the representations of real-world objects or abstract concepts. Tim Berners-Lee expressed these best practices in four principles, known as the Linked Data principles.[4]

- Use URIs as names for things.
- Use HTTP URIs, so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards
- Include links to other URIs, so that they can discover more things.

URIs (Universal Resource Identifier) are used to identify real-world objects and abstract concepts. According to the second principle, information about the entity that the URI represents can be retrieved using HTTP protocol (so-called URI dereferencing). Based on the third principle, which advocates for a standard structure of data dereferenceable by URI, the Resource Description Framework (RDF) has been designed. Tim Berners-Lee also suggested 5-star deployment scheme for ranking published data according to the format in which it is published, comparing them based on their ability to be machine processed. It assigns one star to any data published and assigns the highest number of five stars to data, that is published as RDF, where the entities described are identified by an dereferenceable URI string and the data are connected to other data sources.

1.1 Resource Description Framework

RDF is a data model based on representing data as directed graphs. The basic building block of RDF structured data is a triple consisting of three parts called subject, predicate, and object. The subject is the URI representing the described resource. The object is either URI or literal value like string or number. The predicate specifies the type of relationship between the resources at the positions of subject and object. The predicate is always identified by URI. Predicate URIs come from vocabularies, intended to encompass various relations and concepts occurring in a certain domain.

Set of triples then establishes a RDF graph. URIs at the subject and object positions of the triples make nodes of the graph and each triple acts as an arc connecting the nodes. Type of the connecting is expressed by the predicate URI in the triple. Given the uniqueness of the URIs and their capability of being dereferenced and connected to URIs from various sources (the fourth Linked Data principle), one can imagine the linked data as one giant undivided

graph containing data from various topical domains, so-called Web of Data.

It is important to distinguish the model itself from its formats. RDF describes only an abstract structure of the data that has to be materialized into a certain format when the data is published on the Web. The first standard serialization format published together with RDF in [16] is RDF/XML. An example of two triples described in RDF/XML format is shown in the listing 1.1. The RDF/XML format suits well the use cases, where little human interaction with the data is expected because its syntax is difficult for a human to read and write compared to other formats. On the other hand, its XML background makes it a perfect format for data processed and generated solely by machines.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/">
    <rdf:Description rdf:about="http://example.com/john-johnson">
        <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
        <foaf:name>John Johnson</foaf:name>
    </rdf:Description>
</rdf:RDF>
```

Listing 1.1: Example of RDF data described in RDF/XML format

One of the most used and most human-readable formats is Turtle (Terse RDF Triple Language).[3] It provides various shorthands, enabling to make the representation as brief as possible and thus suitable to be written by hand. The common part of URI strings can be prefixed, so only the decisive end of the URIs has to be stated. The symbol of the semicolon is used to divide pairs of predicate and object belonging to the same subject, so the subject does not have to be repeated. If the described triples share both subject and predicate, a comma can be used to divide the different objects of the triples. Usage of a prefix and the two symbols is shown in an example in the listing 7.5.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix eg: <http://example.com/> .
eg:john-johnson rdf:type foaf:Person ;
                        foaf:name "John Johnson" ;
                        foaf:knows eg:john-jackson, eg:jack-johnson .
```

Listing 1.2: Example of RDF data described in Turtle format

Same as with the relational data model and SQL, RDF also needs a capable language for querying and manipulating the data. For this purposes, SPARQL was designed.[17] Example of a simple SELECT query written is SPARQL is shown in the listing 7.8. Similarly to SQL, the WHERE clause serves to limit the search place from which the result of the query is given. Content of the WHERE clause resembles the Turtle syntax. URIs, that can be bound to variable that occurs in the pattern stated in the WHERE clause, are contained in the output of the query if the variable is enumerated after the SELECT keyword. This

particular query would return all possible bindings for variable `name` ie. all names of persons, for whom the queried data states, that they know a certain John Jackson.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX eg: <http://example.com/>

select ?name where {
    ?person rdf:type foaf:Person ;
            foaf:name ?name ;
            foaf:knows eg:john-jackson .
}
```

Listing 1.3: Example of a simple SPARQL query

1.2 Linked Open Data

The first activities with the goal of starting the publication of Linked Data on a global scale were conducted by the Semantic Web research community as part of the W3C Linking Open Data (LOD) informal initiative established in 2007.[11] The aim of the initiative was to identify datasets published under an open license and to publish them according to the Linked Data principles. All data sets that are published under an open license and are connected to other data sets are referred to as LOD cloud.

The content of the LOD spans across multiple domains. The website lod-cloud.net tracks the current state of published LOD data sets and divides the data sets into these categories, so-called subclouds: Cross-Domain, Geography, Government, Life Sciences, Linquistics, Media, Publications, Social Networking and User-Generated. A data set can fall into more than one category. Cross-Domain, general knowledge data sets play an important role of an intermediary through which unrelated data sets can be connected.

1.2.1 Wikidata

One of those data sets is Wikidata. It is a sister project of Wikipedia, founded and hosted by the Wikimedia Foundation. The data set is managed in an open and collaborative way. Everybody who is interested in expanding the knowledge base can create an account and start contributing. The website of the project provides an intuitive user interface for editing and creating data, so no technical skills beyond common usage of the Internet is needed. The data set currently contains over 93 million items edited by over 26 thousand active contributors. Every item of the dataset is allocated an unique identifier prefixed by the letter **Q**, so-called QID or Q number. The items are described by their statements corresponding to RDF triples. Predicates are in the context of Wikidata called properties and are prefixed by letter **P** similar to items. A sample of the triples contained in Wikidata in Turtle syntax shows listing 1.4.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix wd: <http://www.wikidata.org/entity/> .
@prefix wdt: <http://www.wikidata.org/prop/direct> .
@prefix schema: <http:schema.org/> .

wd:Q111 wdt:P361 wd:Q7879772
       rdf:label "Mars" ;
       schema:description "fourth planet from the Sun";
```

Listing 1.4: Wikidata content sample

A different approach is taken by the DBpedia project. Instead of relying on manual contribution of volunteers, DBpedia's data comes from an application of NLP extraction algorithms over plain text of Wikipedia's articles.

1.2.2 YAGO

This knowledge combines multiple sources of LOD. It contains data about movies, people, cities, countries, etc. The main source of the entities and their relations is Wikidata, but in YAGO all its entity identifiers are converted into a human-readable form and annotated by the *schema.org*[1] ontology maintained by Google. The authors of YAGO call it *a simplified, cleaned, and "reasonable" version of Wikidata.*[24]

The YAGO data set is released in individual versions, unlike Wikidata, which is worked on continuously. The current version is YAGO 4. [22]. All previous versions are available as file dumps[2] and the current version is also accessible via SPARQL endpoint[3].

[1] https://schema.org/
[2] https://yago-knowledge.org/downloads
[3] https://yago-knowledge.org/sparql

2. Data Cubes

While relational databases with highly normalized data models fit well to situations where data is frequently modified, they can be quite cumbersome when being performed complex aggregating queries. Online Analytical Processing (OLAP) system fits better these purposes.

2.1 Terminology

In an OLAP system, numerical data is stored in a multidimensional data structure. The structure is comprised of hypothetical cells, called *observations*, which are identified by their assigned set of *dimension* values from each dimension of the structure. Each observation can contain zero to many numerical values, so-called *measurements*. The meaning of the measurements is referred to as *measure*. In the OLAP context the units of measure are called *attributes*. The structure is referred to as OLAP Cube or Data Cube. The word *cube* implies exactly three dimensions, but its purpose is only to illustrate the multidimensionality of the structure.

Distinct values of a dimension can be organized into a hierarchy, where the parent value is assigned to summarized measurements throughout its child values for each measure. An example of such a hierarchy could be a relationship of a product category and specific products belonging to this category. The depth of a dimension value in its hierarchy then determines the level of granularity the measurement values in a cell assigned to the dimension value are associated with. A cell with all dimension values at the lowest level in their hierarchies or in no hierarchy at all has the finest level of granularity. In a Data Cube consisting of only one cell, meaning each dimension of the cube has only one distinct value, the cell has the coarsest level of granularity.

2.2 Operations

Several operations can be performed on a Data Cube:

roll-up This operation aggregates data either by reduction of one or more dimensions or by climbing up a concept hierarchy for a dimension.

drill-down This operation transforms data to a more detailed level. It is the opposite of roll-up operation. Either a new dimension is added or the values are projected on a more granular level of a dimension.

slice and dice By slicing a cube only certain subset of the dimension values of one dimension is allowed in the resulting cube. Dicing means restricting dimension values across multiple dimensions.

pivot Pivoting means rotating the cube by its axis in order to change the view of the data.

If the values contained in the cube have an additive character (e.g. sales amount or a number of security incidents), the values can be rolled up or drilled down along any dimension. Not all facts are additive though (e.g. average temperature). The analytical process itself lies in performing the above-mentioned operations in order to find interesting insight into the data. By precomputing the aggregations of all possible subsets of dimensions from the cube on the finest level of granularity, the whole process can be accelerated.

2.3 The Data Cube Vocabulary

The principle of dimensions, measures and attributes are the basic building blocks of the standards and guidelines presented by the SDMX (Statistical Data and Metadata eXchange) initiative, that tries to standardize and modernize the exchange of statistical data. The World Wide Web Consortium's recommendation for representing multi-dimensional data in RDF is the Data Cube vocabulary.[6] This vocabulary underlies the standards and guidelines of the SDMX initiative. It allows to publish the content of the cube together with information about its structure and its metadata. The structure of the vocabulary is shown in the picture 2.1.

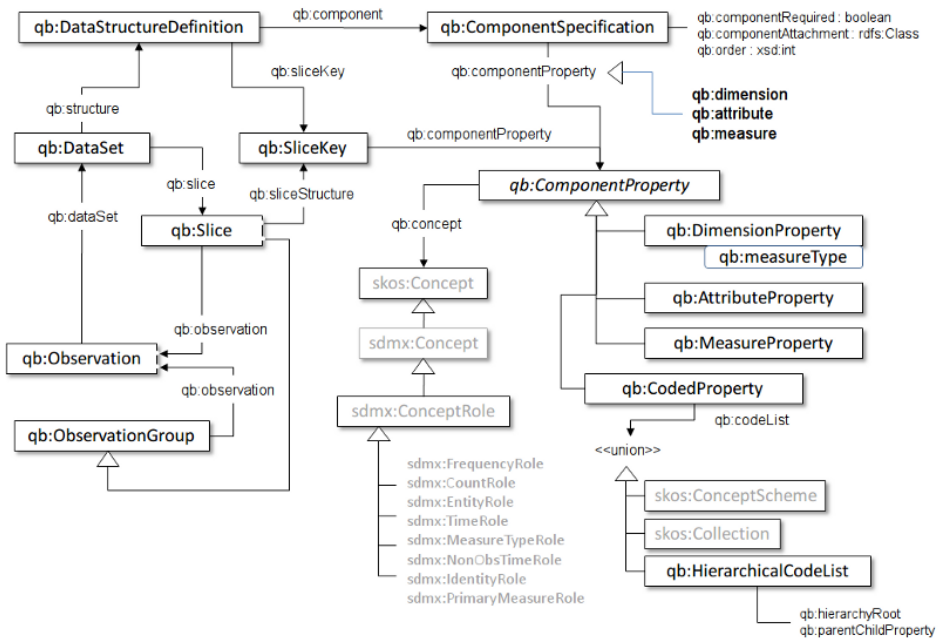

Figure 2.1: The Data Cube Vocabulary structure (Source: [6])

The listing 2.1 shows an example of an observation represented with the Data Cube Vocabulary. The observation has a property $qb : dataSet$ which links it to the entity of the whole cube. The observation is assigned two measures $measure1$ and $measure2$ and is associated with dimension values of two cube's dimensions.

When the cube has to capture more than one measure, the observations can be either structured as in the listing 2.1 or each measurement of a measure can be associated with a different

```
@prefix qb: <http://purl.org/linked-data/cube#> .

<o1> qb:dataSet <dataset1> ;
    <dimension1> <value1> ; <dimension2> <value2> ;
    <measure1> 12030 ;
    <measure2> 3 .
```

Listing 2.1: Data Cube Vocabulary observations example 1

observation. Those observations than share the dimension values.

```
@prefix qb: <http://purl.org/linked-data/cube#> .

<o1> qb:dataSet <dataset1> ;
    <dimension1> <value1> ; <dimension2> <value2> ;
    <measure1> 12030 ;

<o2> qb:dataSet <dataset1> ;
    <dimension1> <value1> ; <dimension2> <value2> ;
    <measure2> 3 .
```

Listing 2.2: Data Cube Vocabulary observations example 2

2.4 Simple Knowledge Organization System

Simple Knowledge Organization System (SKOS) is a model and an RDF vocabulary for expressing the basic structure and content of concept schemes.[1] It is the most reused LOD vocabulary for the representation of code lists and hierarchies. That is why it is especially fitting for representing possible dimension values. There are two main building blocks in SKOS. The $skos : ConceptScheme$ class represents the code list itself. The $skos : Concept$ class represents the individual code list items.

```
@prefix eg: <http://example.com/skos#> .

eg:thingsILike a skos:ConceptScheme ;
    skos:prefLabel "Things I like"@en .

eg:Food a skos:Concept;
    skos:prefLabel "food"@en ; skos:inScheme eg:thingsILike ;
    skos:notation "FOOD" ; skos:altLabel "jídlo"@cs .

eg:Sleep a skos:Concept;
    skos:prefLabel "sleep"@en ; skos:inScheme eg:thingsILike ;
    skos:notation "SLEEP" ; skos:altLabel "sánek"@cs .
```

Listing 2.3: Example of a SKOS concept scheme

3. Association Rules

Association rules are one of the machine learning methods. The method lies in finding often repeating constructions in the analyzed data, which is mostly in the form of a table, where each row represents a *transaction* which is described by values in the table's columns.[2] A found association rule states that the transactions with a certain set of properties, called *antecedent*, are associated with a different set of properties called the *consequent*.

$$antecedent \implies consequent$$

An example of an association rule can be a statement, that the customers of a bookstore often buy a book from the self-help section together with a book from the esoteric section. Such rule can be found in a table where each row represents one order of a customer in the bookstore, each column represents a book available in the store, and the values express whether the particular book is a part of the particular order.

3.1 Interest Measures

The relation between the antecedent and consequent in the analyzed data is often represented in the form of a four-field table. It is a *2 x 2* matrix, where the rows represent the antecedent and its negation and the columns represent the consequent and its negation. The numerical values in the matrix state the number of transactions corresponding to the properties in their row and column. These values are denoted as *a, b, c, d*.

	Consequent	**¬ Consequent**
Antecedent	a	b
¬ Antecedent	c	d

Table 3.1: Four-field table of association rule

The strength of the association is expressed by the *interest measures*, also called the *measures of significance*. The found association rules can be compared based on those interest measures. The interest measures are computed by a formula from the values in the rule's four-field table.

Confidence states the ratio of the number of transactions satisfying both the antecedent and the consequent of the rule over the number of transactions that satisfy the antecedent of the rule.

$$confidence = \frac{a}{a+b}$$

Support is the number of transactions satisfying both antecedent and consequent i.e the transaction for which the rule is valid. It is also possible to define it as relative support,

where the number of valid transactions is divided by the number of all transactions in the data.

$$relSupport = \frac{a}{a+b+c+d}$$

Lift represents the degree by which the probability of the right prediction of the set of properties in the consequent is improved by the validity of the antecedent in a transaction.

$$lift = \frac{\frac{a}{a+b}}{\frac{a+c}{a+b+c+d}} = \frac{a*(a+b+c+d)}{(a+b)*(a+c)}$$

Coverage represents the conditional probability that the antecedent of the rule is valid given that the consequent is valid for the transaction. In other words, it expresses the ratio of positive examples in the data *covered* by the rule.

$$lift = \frac{a}{a+c}$$

4. AMIE Algorithm and Its Derivatives

Finding association rules in knowledge bases can serve several purposes. New facts that are not yet present in the dataset can be derived from the found regularities described by the rules. From such rules opposing facts present in the dataset can be deduced to be wrong. Mined rules can also help to understand the data better.

For mining rules from a graph database such as LOD datasets, Inductive Logic Programming (ILP) can be used. ILP works under the Closed World Assumption (CWA) meaning it supposed that both negative and positive statements are present in the data. However LOD operates under the Open World Assumption (OWA) ie. if a statement is not present in the data, it does not mean that this statement does not correspond to reality. Rules mined by ILP would not reflect this matter. Moreover, ILP are not observed to be efficient over large datasets in the order of millions of statements, making it not a viable way to mine rules over real-world knowledge bases such as YAGO or Wikidata.

4.1 AMIE

An algorithm that is specifically designed to mine rules from data operating under OWA and consisting of binary predicates (just as Linked Data) is AMIE.[8] AMIE mines rules in the form of Horn rule. Horn rule is an implication with conjunction of atoms on the left side, called body and a single atom on right side call head. We can imagine an atom as an RDF triple, where subject and object can be replaced by variables. The number of atoms in a rule indicates the length of the rule. In this work, rules are represented with an infix notation. An example of a rule AMIE seeks to discover is shown below.

$$(?a\ worksIn\ ?b) \wedge (?b\ hasHeadquartersIn\ ?c) \Rightarrow (?a\ livesIn\ ?c)$$

This rule states that any person lives in a place his or her company's headquarters. The length of this rule is 3 since it has two body atoms. The rule has 3 variables. When we substitute the variables by constants present in the examined data set, we get an *instantiation* of the rule. If all atoms of the instantiated rule appear in the data set, the head atom of the instantiation is one of the *predictions* of the rule. The number of all instantiations of an atom that appear in the data set is called *size* of the atom.

4.1.1 Language Bias

In order to efficiently traverse the search space, AMIE subjects the rules to a particular language bias. Only the rules conforming to the conditions stated below can be generated

and further refined.

rules have to be connected A rule is connected when every atom in the rule shares every variable with another atom in the rule.

rules have to be closed A rule is closed when every variable appears at least twice in the rule.

rules cannot be reflexive reflexive rule contains at least one atom with identical subject and object variable or constant.

rule can be recursive Any predicate can occur more than once in a rule.

4.1.2 Measures of Significance

Support

For a chosen definition of a support measure for the AMIE algorithm, it is crucial for the definition to have the property of monotonicity ie. by adding any new atom to the body of a rule, the support of the rule shall always decrease or remain the same. A naive way to count support of a rule would be to count all instantiations of the rule that appear in KB. Such definition would not comply with the property of monotonicity, since the addition of a dangling atom to a rule would introduce a new variable multiplying the number of instantiations and thus the value of the support measure. By counting only all distinct pairs of subjects and objects in the head of all instantiations that appear in KB, the property of monotonicity is preserved:

$$supp(\vec{B} \Rightarrow H) = \#\langle s,p,o\rangle \in KG : \exists t_1,...,t_n \in KG : ((t_1 \wedge ... \wedge t_n) \Rightarrow \langle s,p,o\rangle) \prec (\vec{B} \Rightarrow H)$$

Head Coverage

Since the support is an absolute number, so the size of the examined data set has to be taken into account while defining this threshold. Plus if the defined support value is greater than a number of distinct triples containing a certain predicate, any rule containing this predicate in the head atom would be disregarded. Head Coverage is the relative expression of support. It is defined as support of a rule over the numbers of triples with the head's predicate, so-called *head size*.

$$hc(\vec{B} \Rightarrow H) = \frac{supp(\vec{B} \Rightarrow H)}{hsize(\vec{B} \Rightarrow H)}$$

4.1.3 Confidence Measures

The above-mentioned measures describe a quantitative significance of the rule in relation to the examined data set. They quantify the true predictions of the rule but do not take into account the false predictions. Confidence is a way to measure the quality of a rule. Generally speaking, confidence is a ratio of true predictions of a rule to the sum of true predictions and the counterexamples. The number of true predictions can easily be expressed by the rule's support. Two different ways to count the counterexamples are discussed below.

CWA and Standard Confidence

Standard confidence considers every fact that is not present in the examined dataset a false fact and thus a counterexample when predicted by a rule. A fact predicted by a rule is either present in the data set or is not. Therefore the standard confidence is defined as the ratio of the number of true predictions of the rule to the number of all predictions of the rule.

$$conf(\vec{B} \Rightarrow H) := \frac{supp(\vec{B} \Rightarrow H)}{bsize(\vec{B} \Rightarrow H)}$$

$$bsize(\vec{B} \Rightarrow H) = \#\langle s,p,o\rangle : \exists t_1,...,t_n \in KG : ((t_1 \wedge ... \wedge t_n) \Rightarrow \langle s,p,o\rangle) \prec (\vec{B} \Rightarrow H)$$

This way of generating counterexamples fails to distinguish a false fact from an unknown fact. This conforms to CWA and it is traditionally used for association rule mining over transactional data where this assumption can be applied. For example, if the data does not state, that I bought a bottle of milk last Wednesday, then I really did not buy it. AMIE, however, is intended to mine rules from data operating under OWA, so the usage of this measure is inappropriate.

PCA Confidence

For the PCA Confidence, Partial Completeness Assumption (PCA) is used for generating the counterexamples:

If $\langle s\ p\ o\rangle \in KBtrue$ then $\forall_{o'} : \langle s\ p\ o'\rangle)) \in (KBtrue \cup NEWtrue) \Rightarrow \langle s\ p\ o'\rangle \in KBtrue$.

Meaning that if we know any object for a given predicate and subject, we know all triples containing the predicate and subject together. This assumption is certainly true for predicates with high or complete functionality, such as birthdate or capital. A triple predicted by the measured rule is considered a counterexample only when triples with its combination of subject and predicate are present in the data set and none of those has the triple's object.

$$conf_{pca} := \frac{supp(\vec{B} \Rightarrow H)}{bsize_{pca}(\vec{B} \Rightarrow H)}$$

$$bsize_{pca}(\vec{B} \Rightarrow H) = \#\langle s,p,o\rangle : \exists t_1,...,t_n,t' \in KG :$$
$$((t_1 \wedge ... \wedge t_n) \Rightarrow \langle s,p,o\rangle) \prec (\vec{B} \Rightarrow H) \wedge t' = \langle s',p,o'\rangle$$

4.1.4 Algorithm

The AMIE algorithm takes as input parameters the RDF graph the rules are to be mined from (KG), the minimum head coverage of the rules ($minHC$), maximal length of the rules ($maxLen$) and the minimal confidence of the rules ($minConf$). For each distinct predicate in the graph, a rule with an empty body and a head atom with this predicate is created and the algorithm is initialized by filling a queue with those rules.

The algorithm then iteratively dequeues rules from the queue. If the dequeued rule complies with the criteria set by the input parameters, it is accepted for output and added to the result rules array. If the rule's length is shorter than the maximal stated length, the rule is refined meaning new atoms are added to its body to create new rules. If a new rule reaches the defined minimal head coverage and is not already present in the queue, it is added to the rule queue. The algorithm continues dequeuing the rules until the queue is empty and no more new rules can be created by the refinement.

Algorithm 1 AMIE algorithm

```
1: procedure AMIE(KG, minHC, maxLen, minConf)
2:     queue = [(?a r1 ?b), (?a r2 ?b) ... (?a rm ?b)]
3:     output = ⟨⟩
4:     while ¬queue.isEmpty() do
5:         rule = queue.dequeue()
6:         if AcceptedForOutput(r, out, minConf) then
7:             output.add(rule)
8:         end if
9:         if length(rule) < maxLen then
10:            R(rule) = Refine(rule)
11:        end if
12:        for ri ∈ R(rule) do
13:            if hc(ri ≥ minHC & ri ∉ queue) then
14:                queue.enqueue(ri)
15:            end if
16:        end for
17:    end while
18:    return output
19: end procedure
```

When Choosing larger thresholds on minimal head coverage and minimal confidence and shorter maximum length makes the algorithm stop earlier and generate fewer rules. Choosing smaller thresholds and allowing the generation of longer rules results in longer runtime and more found rules. So the setting of these parameters always introduces a trade-off between the runtime and the number of returned rules.

4.1.5 Rule Refinement

By simply enumerating all possible rules and then computing the interest measures for them in order to find the significant ones is not a feasible approach for large graphs. The exploration of the search space of the rules has to be done efficiently. Such exploration is performed by the rules refinement. During the rule refinement, the new atoms are added to the rule by three means (called *operators*). Each operator creates zero to many rules when applied to a rule.

O_D A *dangling* atom is added to the rule. This atom introduces a new variable to the rule. Its second variable is already present in the rule.

O_I An *instantiated* atom extends the rule. An instantiated atom contains a constant and a variable already present in the rule.

O_C A *closing* atom is added. This atom's variables are already present in the rule.

4.1.6 Querying the Graph

The algorithm uses so-called *count projection queries* to find the predicates and entities with which new atoms are created during the rule refinement by applying the mining operators, such that the minimum head coverage of the new rule is reached. These queries are not efficient when implemented in SPARQL or SQL. So the authors of the AMIE algorithm suggested an in-memory database that is tailored to this type of queries.

The data structure consists of six *fact indices*: each for a permutation of the subject (`S`), predicate (`P`), and object (`O`). Let them be denoted as `SPO`, `SOP`, `PSO`, `POS`, `OSP` and `OPS`. Each fact index is a hash table that maps elements of the first column to a hash table that contains elements of the second column as key mapping to a set of third column elements, such that triples in the graph containing the first column element, second column element and the third column element exist. For example the index SPO is a hash table with each subject present in the graph as a key. Every subject key points to a hash table with keys of predicates that exist with this subject in at least one triple in the graph. The predicate key then points to an array of objects for which triples with this subject, predicate, and object exist in the graph. That means that each triple in the graph is store six times in this database.

Along with the fact indices, the *aggregated* indices are used for the algorithm. There are three (`S`, `P`, `O`) of them for each triple position and they store the number of triples in the graph that contain the element of the corresponding key. For example, the index `P` stores number of triples for each distinct predicate in the graph. The numbers have to be precomputed before initializing the mining. With these indices, it is easy to check of size of an atom (*size queries*). When computing for example the size of atom `?a knows ?b` the algorithm would turn to the aggregated index P and simply look up the number corresponding to the key `knows`. For an atom with a constant instead of one of the variables, a fact index is used. For computing the size of the atom `?a knows John_Smith` the algorithm would look into the `POS` index and find

a hash table corresponding to the predicate `knows`. Then in this hash table, it would find the array corresponding to the key of the entity `John_Smith` and count the items in the array. For checking the existence of a triple any fact index can be used. Either way, the database allows checking for the existence of a triple or computing atom's size in constant time. A drawback of this is that the database is six times more memory-demanding than the input graph itself.

The algorithm exercises other types of queries. The size queries are part of the so-called *existence queries* that check for the existence of a binding of conjunction of atoms. The existence queries are in turn used in the so-called *select queries* that look for all instances of a variable in conjunction of atoms. The select queries are part of the *count queries* that count distinct bindings to variables of an atom in conjunction of atom. This is useful for computing the confidence of a rule.

For the PCA confidence of a rule the denominator is the bindings count of head atom's variables x and y for which $(x\ r\ y') \wedge \vec{B}$ can be instantiated from the graph. To compute the denominator, the algorithm first fires SELECT DISTINCT x where $(x\ r\ y') \wedge \vec{B}$. Then for each found x it instantiates the conjunction with with x and fires SELECT DISTINCT y where $(x\ r\ y') \wedge \vec{B}$. By adding up the numbers of distinct y from each query the denominator is computed.

Count Projection Queries

The select queries are also necessary for the count projection queries to find the predicates and entities for new atoms for a rule during the rule refinement. The algorithm considers the set minimal head coverage when creating the new atoms to be added. Only the atoms are added for which the new rule still reaches the minimal head coverage. The general structure of a count projection query is as this:

$SELECT\ x, COUNT(H)\ WHERE\ H \wedge \vec{B}$
$SUCH\ THAT\ COUNT(H) \geq k$

where x can represent either predicates or entities of new atoms. $\vec{B}$ is body of the new rule including the added atom. The symbol k stands for an absolute translation of the minimal head coverage for the rule: $minHC \times size(H)$. The authors of the algorithm provide following example for explaining the usage of count projection queries during a rule refinement. The rule $(?x\ marriedTo\ ?z) \Rightarrow (?x\ livesIn\ ?y)$ is about to be refined. When applying a dangling atom operator to this rule, atoms are to be created that contain a new variable $(?w)$. The new variable in the new atoms will be either:

- at the position of subject and the variable $?y$ will be at the position of object: $(?w\ p\ ?y)$
- at the position of subject and the variable $?z$ will be at the position of object: $(?w\ p\ ?z)$
- at the position of object and the variable $?y$ will be at the position of subject: $(?y\ p\ ?w)$
- at the position of object and the variable $?z$ will be at the position of subject: $(?z\ p\ ?w)$

For each of those four *join combinations*, a count projection query will be fired to find predicates to be substituted into them:

$SELECT\ p, COUNT(?x\ livesIn\ ?y)$
$WHERE\ (?x\ livesIn\ ?y) \wedge (?x\ marriedTo\ ?z) \wedge (X\ p\ Y)$
$SUCH\ THAT\ COUNT(?x\ livesIn\ ?y) \geq k$

$(X\ p\ Y) \in \{(?w\ p\ ?y),(?w\ p\ ?z),(?y\ p\ ?w),(?z\ p\ ?w)\}$

If the dangling atom operator is applied on an intermediate rule (not all variables are closed) like this one, the dangling atoms are joined on the non-closed variables ($?y$ and $?z$) in this case. If the rule is closed, the dangling atom is joined on all variables in the rule. The closed atom operator would apply join combinations of $(?y\ p\ ?z)$ and $(?z\ p\ ?y)$ for this example. If there were only one open variable, the closed atom operator would apply join combinations with this variable and each other variable in the rule. If there were no open variables, the operator would apply join combinations of all possible pairs of variables in the rule. When applying the instantiated atom operator, first the dangling atom is used to generate new atoms with new variable. For each this new atom a count projection query on instances of the new variable in the atom is fired to find all entities, that can substitute the variable and the minimal head coverage is still reached. Each found entity then forms a new rule with the new atom with the instantiated variable. This is an example of the query for atom $(?x\ citizenOf\ ?w)$:

$SELECT\ w, COUNT(?x\ livesIn\ ?y)\ WHERE$
$(?x\ livesIn\ ?y) \wedge (?x\ marriedTo\ ?z) \wedge (?x\ citizenOf\ w)$
$SUCH\ THAT\ COUNT(?x\ livesIn\ ?y) \geq k$

4.2 AMIE+

In [7] the authors of the AMIE algorithm came up with an improved version referring to it as AMIE+. The improvements lie in the rule refinement phase and the confidence evaluation. The improvements do not alter the output of the algorithm, they only speed it up so that it is applicable to mining over large knowledge bases.

4.2.1 Rule Refinement

From the refined rule, it is sometimes possible to infer that an application of an operator would not yield better on any new rules. AMIE+ adds these checks before an operator application:

1. When a not-closed rule with a length of $maxLen - 1$ is being refined, the dangling atom operator is not applied, since the new variable cannot be closed before exceeding the $maxLen$ threshold.

2. When a not-closed rule with a length of $maxLen - 1$ and more than two open variables is being refined, the closed atom operator is not applied, because one variable would still be open when the rule exceeds the $maxLen$ constraints so no such rules would be accepted for output.
3. A not-closed rule with a length of $maxLen - 1$ and more than one open variable is not applied an instantiated atom operator on, because the instantiated atom operator can only close at most one variable.
4. If the refined rule reaches PCA confidence of 1, it is no further refined, since the new rule would not increase in confidence. They can only decrease in support.

Also in some cases, a dangling atom cannot reduce the support of a rule. It is when the parent rule already contains atoms with the same relation as a dangling atom and these atoms have a variable in common with the dangling atom. The child rules would have the same support as the parent rule, so the support computation for these rules can be skipped.

4.2.2 Speeding Up Confidence Evaluation

The most important improvement is the confidence approximation. The authors state that during experiments with AMIE, 35% of the runtime was spent on the confidence computation. The algorithm has to at first compute the rule's confidence and only after that the rule can be discarded when it does not reach a $minConf$ threshold. So the algorithm can spend hundreds of milliseconds computing the confidence of a rule for nothing. But instead of computing it, the confidence can be estimated, which is much faster (the authors claim 200-fold speed up). The approximation is based on statistics about predicates (functionality and inverse functionality of predicates, size of the joins between predicates, etc.) in the input graph, that are precomputed when the input graph is loaded into the in-memory database.

The approximation is designed to overestimate the confidence so that no high confidence rules are pruned. But the approximation does not just simply substitute the exact confidence computation. The approximation is only applied to rules that have intermediate variables (variables that do not appear in the head atom) and there exists a single path to one head variable to the other through the intermediate variables. For example the rule $(?a\ livesIn\ ?b) \wedge (?b\ hasStreet\ ?c) \Rightarrow (?a\ worksIn\ ?c)$ has one intermediate variable $?b$. There is a single path of variables connecting the variables of the rule's head: $?a \rightarrow ?b \rightarrow ?c$. So this rule's confidence would be approximated instead of computed. Rules without intermediate variables are supposed to have a smaller number of bindings to their body so the confidence should be quicker to compute. Confidence for rules with more than single path connecting the head variables (such as $(?a\ livesIn\ ?b) \wedge (?b\ hasStreet\ ?c) \wedge (?a\ bornIn\ ?d) \wedge (?d\ hasStreet\ ?c) \Rightarrow (?a\ worksIn\ ?c)$ that has two paths: $?a \rightarrow ?b \rightarrow ?c$ and $?a \rightarrow ?d \rightarrow ?c$) is also thought to be easy to compute because more variable paths tend to restrict the number of head variables bindings. If the estimated value reaches the $minConf$ threshold, the exact confidence value is computed as well.

4.3 RDFRules

Although the AMIE+ algorithm made it possible to mine rules from large knowledge bases, its use would have to be combined with other tools when deployed on real-world knowledge bases, since it does not deal with preprocessing the input data (e.g. unifying IRIs for identical resources, treatment of numerical values) and subsequent analysis of the generated rules. An extension of the AMIE+ algorithm under the name of *RDFRules* was presented in [26]. Beside enhancements to the algorithm itself (faster projection counting pruning by used specified *rule patterns*), several other techniques were proposed to be integrated with the algorithm to cover the complete mining process.

4.3.1 Processing of Numerical Attributes

AMIE+ treats numerical values (in RDF those can only appear at the position of object) just as any other discrete value. Due to the downward closure property of the algorithm (and of the association rule mining algorithms in general), where not only the whole rule but also each subset of the rule's atom have to meet the minimum support threshold, and due to the fact that the numerical attributes usually contain many distinct values, the rules concerning numerical values tend to be excluded from the generated rules.

This problem can be solved by the values' discretization meaning that the continuous values are converted to a finite set of intervals. When working with transactional data such sets of intervals are created within each examined table's column which contains numerical values. For example, the *arules*[1] R package implementing the Apriori algorithm enables discretizing[2] the values either equidistantly (the intervals have a given fixed length), equifrequently (a given number of intervals are evenly represented in the continuous data points), by clustering the values to a given number of intervals, or by stating the intervals manually.

In the context of RDF, the discretization entails grouping numerical values in the range of a certain predicate. This can also be done equifrequently, equidistantly, etc. However, when the created intervals are too narrow, the rules containing the discretized predicates may not satisfy the minimum support or the head coverage threshold defined for the mining task. When creating too broad intervals the found rules may be unnecessarily general. That means that the discretization phase cannot easily be decoupled from the mining phase itself.

In [26] the authors propose a discretization heuristic that takes into account the minimum head coverage and minimum head size thresholds to perform a specific discretization for a particular task in the pre-processing phase. For each predicate in the data set that has a numerical range a set of overlapping intervals is generated. For each interval in this set and for each predicate's triple with the number in the range of the interval a copy of the triple

[1] https://www.rdocumentation.org/packages/arules/versions/1.6-8
[2] https://www.rdocumentation.org/packages/arules/versions/1.6-8/topics/discretize

whose object is substituted and whose predicate is altered by a suffix of the interval set (not to modify the range of the original predicate) by the interval is added to the data set.

4.3.2 Multiple Graphs

Restricting the mining task onto a single graph prevents finding relations across multiple sources. Even though the resources representing the identical object or a concept in the real world have a different identifier in different graphs, the identity can be easily inferred from the `owl:sameAs` predicate joining two identical resources.

$$(?a\ wasBornIn\ ?b\ YAGO) \Rightarrow (?a\ dbo:deathPlace\ ?b\ DBpedia)$$

AMIE+ does not recognize the `owl:sameAs` statements and assumes that the identical resources have the same IRI. To mine from across multiple graphs the graph would either have to be merged together and their IRIs would have to be unified or the `owl:sameAs` relations would have to be explicitly stated in the rules.

RDFRules natively supports *quads* i.e triples enriched with IRI of their corresponding named graph. On top of the AMIE's six indices, four new ones are added (**PG**, **PSG**, **POG**, **PSOG**) that allow to check the affiliation of triples to a graph. Not only the individual triples but also the atoms in a rule can be treated as quads. The atoms can be restricted via the rule patterns (see 4.3.3) to be based only on triples from a certain graph. The RDFRules algorithm recognizes the `owl:sameAs` triples and the resources joined this way are treated as if they had identical identifiers.

4.3.3 Improvements to Expressiveness of Rule Patterns

The association rules mining algorithms tend to create rules describing obvious and uninteresting patterns when the search space of the rules is not properly restricted. AMIE allows the user to provide a list of predicates that should be included or excluded in the rules' body and head and also to prohibit or enforce constants, so that the shape of the rules can be controlled to some extent. RDFRules offers a much more profound solution in the form of a rule pattern grammar.

The user defines rule patterns that are used to prune the rules during the rule refinement. Each rule has to match at least one of the rule patterns defined for the mining task. A rule pattern consists of atom patterns corresponding to atoms of a matching rule. The rule pattern can have zero or any number of atom patterns in the body and exactly one atom pattern in its head. A rule pattern with atom patterns in the body and no atom pattern in the head is useless since the rule patterns are applied from right to left just as the rule refinement operators. An atom pattern consists of four atom item patterns that correspond

to the subject, predicate, object, and graph of the matching rule atom. Those atom item patterns are available:

? pattern for any symbol at the position
$?_v$ pattern for any variable
$?_c$ pattern for any constant
$?a$ pattern for a concrete variable written as a single alphabetic character after the symbol ?
[ ] collection of items where at least one has to be matched at the position, that can be also negated by the symbol ¬ stating that none of the items must match at the position

Shown below is an example of a valid rule pattern and its matching rule:

$$(?a\ rdf:type\ ?_c) \wedge (?a\ ?\ ?_v) \Rightarrow (?a\ ?\ ?)$$

$$(?a\ rdf:type\ dbo:Scientist) \wedge (?a\ dbo:academicDiscipline\ ?b) \Rightarrow (?a\ dbo:knownFor\ ?b)$$

4.3.4 Top-K Approach

RDFRules algorithm offers an alternative to manually defining an interest measure threshold (e.g. minimum support or minimum confidence) for the mining task. Instead, in the so-called *top-k approach*, the user defines a maximum number of rules with the highest values of a chosen measure that should be returned by the algorithm. During the mining process, the rules are stored and sorted in a queue with a fixed length of the defined number. The head of the queue contains a rule with the lowest value of the chosen measure. This value acts as a temporary minimum threshold. Once a rule with a higher value of the measure is found, the head rule is removed from the queue, the new rule is added to the queue and the queue is sorted again so that the head of the queue still contains the rule with the lowest value. At the end of the mining process, only the rules in the queue are returned.

4.3.5 Support for the Lift Measure

The lift is a measure that describes how the probability of the consequent (head) occurrence is increased given that the antecedent (body) of the rule is valid compared to its probability of occurrence under a random choice in the complete dataset. RDFRules adds support for this measure. It is computed as a ratio of the rule's confidence and its *head confidence*.

$$lift(\vec{B} \Rightarrow H) = \frac{conf(\vec{B} \Rightarrow H)}{hconf(H)}$$

The formula of the head confidence depends on the type of the head atom. If the head atom contains two variables ($H = (?a\ p\ ?b)$), the head confidence is computed as the ratio between the number of distinct subjects bound with the predicate in the head atom and the number of distinct subjects in the whole data set.

$$hconf(\vec{B} \Rightarrow H) = \frac{\#s : \exists\langle s,p,o\rangle \prec (?a\ p\ ?b)}{\#s : \exists\langle s,p,o\rangle \prec (?a\ ?r\ ?b)}$$

If the head atom contains one variable and a constant at the position of object ($H = (?a\ p\ C)$), then the head confidence is computed as the ratio between the number of distinct subjects bound with the predicate in the head atom and with the constant in the head atom and the number of distinct subjects in the whole data set.

$$hconf(\vec{B} \Rightarrow H) = \frac{\#s : \exists\langle s,p,o\rangle \prec (?a\ p\ C)}{\#s : \exists\langle s,p,o\rangle \prec (?a\ p\ ?b)}$$

If the head atom contains a constant at the position of subject ($H = (C\ p\ ?a)$), the formula goes as the ratio of between the number of distinct objects bound with the predicate in the head atom and with the constant in the head atom and the number of distinct subjects in the whole data set.

$$hconf(\vec{B} \Rightarrow H) = \frac{\#s : \exists\langle s,p,o\rangle \prec (C\ p\ ?a)}{\#s : \exists\langle s,p,o\rangle \prec (?a\ p\ ?b)}$$

4.3.6 Rule Clustering and Pruning

When an association rule mining algorithm generates a high number of overlapping rules, a clustering algorithm can be used to group the rules, so that the rules can be presented in a more compact and organized way e.g. as a result of an exploratory analysis. In [26] propose an approach to determine the similarity of two rules based on their content and their interest measures. When comparing the content similarity of rules U and V, where $|U| \geq |V|$ the similarity is computed as the maximum average of atom similarities between atoms of the rule V and a k-permutation of the atoms from the rule U where $k = |V|$.

$$sim_c(U,V) = \frac{1}{|V|} \max_{T \in P(U,|V|)} \sum_{i=1}^{|T|} sim_a(t_i,v_i)$$

The similarity of two atoms is based on the similarities between their subjects, predicates and objects.

$$sim_a(A_1, A_2) = \frac{1}{3}[\, sim(\langle s_1, p_1\rangle, \langle s_2, p_2\rangle) + sim(\langle o_1, p_1\rangle, \langle o_2, p_2\rangle) + sim(p_1, p_2)]$$

The similarity function comparing two predicates returns the value 1 if the predicates are identical and 0 otherwise. The similarity function comparing two subjects (and objects analogously) returns the value 1 either if the subjects are identical and they are not variables or if they are both variables and the predicates of the two rules are identical, it returns the value 0.5 if the subjects are not identical, however, the predicates are identical and exactly one of the subjects is a variable, and it returns 0 otherwise.

To avoid a situation where a single triple in the data set is covered by multiple rules returned by the algorithm, the authors of RDFRules suggest adapting the *data coverage pruning* in the post-processing phase. The rules are ranked based on the following criteria:

1. $conf(A) > conf(B)$
2. $conf(A) = conf(B)$ and $hc(A) > hc(B)$
3. rule A has a shorter body than the rule B

Then for each rule starting with the highest ranked to the lowest-ranked rule it is checked whether the rule covers at least one triple (i.e. any atom in the rule can be instantiated by the triple) that the previous rules did not and only those rules which satisfy the conditions are kept. That way the new set of rules covers exactly the same set of triples in the data set.

5. RDFRules Reference Implementation

[26] also presents an open-source implementation of the improved AMIE+ algorithm and some of the proposed supporting techniques[1] as a single framework under the name RDFRules. The source code of the framework is available at https://github.com/propi/rdfrules. It can act as an equivalent to the modern algoritmic frameworks for mining association rules from tabular data, such as *arules* R package or Spark MLlib that enable to deal with the whole mining process with just one tool.

5.1 Interfaces

The core of the framework is written in Scala. Beside the Scala API[2], the framework also provides a Java API[3] serving as the facade into the Scala core, though it is already[4] pronounced deprecated. Both APIs are published[5] in the JitPack package repository, so they can be easily added to a Scala or Java project as a dependency. The framework also has a REST API wrapping the Scala core that can either be used as is and be accessed through an HTTP client or through a GUI via a web broswer.

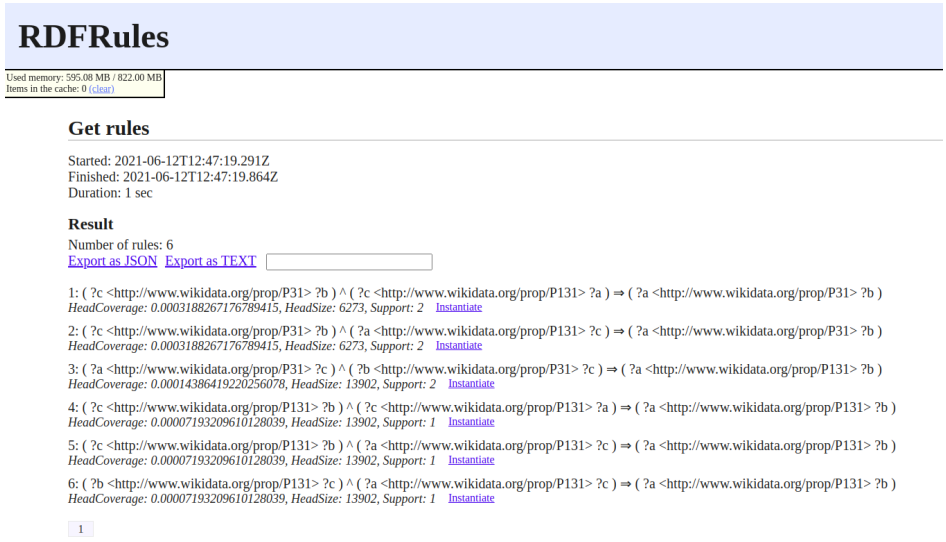

Figure 5.1: RDFRules web browser interface

[1] Unlike the reference implementation of the AMIE+ authors that focuses solely on the modelling phase.
[2] https://github.com/propi/rdfrules/tree/master/core
[3] https://github.com/propi/rdfrules/tree/master/java
[4] The current version of the framework as of writing this is *1.5.0* published on November 28th 2020.
[5] https://jitpack.io/#propi/rdfrules

5.2 Data Structures

The framework's core revolves around four main data structures: *RDFGraph* and *RDFDataset* corresponding to the input data, *Index* wrapping the set of indices the input data is transformed into and *RuleSet* containing rules generated by the algorithm. These structures are transformed into each other in the stated order during the mining process by applying various operations on them. Inspired by the Apache Spark RDD[6] the operations are categorized into *transformations* and *actions*. All transformations are *lazy* operations meaning they are not performed right after the API call but only when an action operation is called which is dependent on them.

Each transformation method called on an instance of the data structures returns either a modified version of the same instance or the instance transform into a further data structure but the instance which the method was called on does not change i.e. the objects of the framework are immutable and the whole workflow with the core API lies in chaining those methods. Multiple calls of an action operation with different input parameters would result in repetitive performance of the defined transformations so each data structure has a *cache* method that enables preserving the result of the transformations in memory or on the disk so that they have to be performed only once.

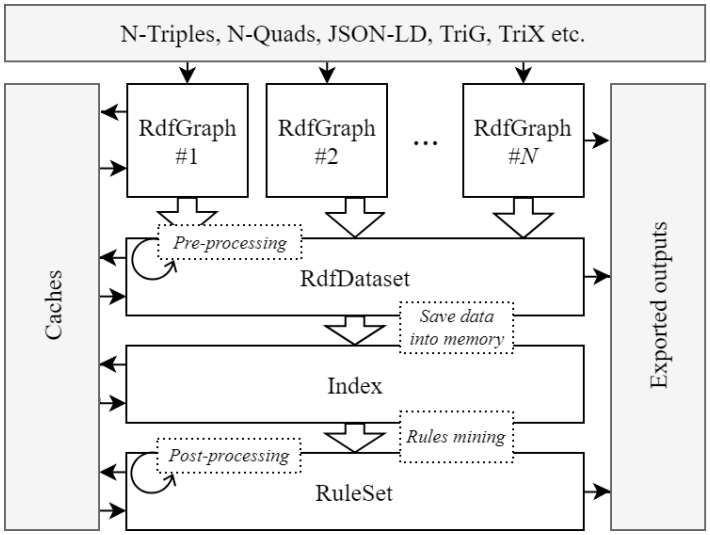

Figure 5.2: Relations between the RDFRules data structures (Source: [25])

An RDFGraph instance corresponds to a set of triples. It is created either from stream of triples or from a file containing a supported RDF serialization (N-Triples, N-Quads, JSON-LD, TriG, Turtle, etc.) and optionally can be assigned a graph IRI. RDFDataset can either be created from one or multiple instances of RDFGraph or directly from a file if the input format contains a set of quads so the RDFDataset corresponds to a set of quads. Both RDFGraph and RDFDataset allow filtering, slicing or modifying the data at the level of individual triples/quads. It is possible to merge two instances of RDFGraph, two instances

[6] https://spark.apache.org/docs/latest/rdd-programming-guide.html#resilient-distributed-datasets-rdds

of RDFDataset and to merge an instance of RDFDataset and RDFGraph which results in a new instance of RDFDataset.

Both have a *discretize* method that facades into the EasyMiner-Discretization[7] library. It takes a parameter specifying the kind of discretization (*discretization task*) and a parameter in a form of a function that specififfies which triples are to be processed. It allows to create a defined number of equidistant and equifrequent intervals or to create equifrequent intervals with a defined minimum frequency (*equisize* discretization task).

The RDFDataset instance is transformed to an instance of Index on which the rules can be mined with. During this transformation, each resource and literal is assigned a unique number which reprents it in the created indices. The mining itself, which is triggered by a *mine* method is controlled by defined *pruning thresholds* (pruning during the rule refinement), rule patterns and *constraints*. The available pruning thresholds are minimal head size, minimal head coverage, minimal support, maximum rule length and *timeout* (a time period after which the mining is stopped and so far found rules are returned) in minutes. It also implements the TopK pruning threshold mentioned in [26] although only for the head coverage. It does not allow pruning the rules based on minimum confidence as the AMIE+ implementation of Galárraga et al. does. The constraints can specify general characteristics of the rules that can be returned e.g. disallowing duplicit predicates or any constant in the rules.

The rules are returned by the *mine* method as an instance of RuleSet. Each rule in the set basic measures such as support, head size, etc. and other *computationaly expensive* measures such as confidence (PCA or a standard confidence) can be calculated optionally. The rules in the set can be filtered and sorted by those measures. Pruning and clustering can be performed on the set. The algorithm used for clustering is DBScan which takes parameters of minimum size of a cluster, minimum similarity of rules in the same cluster and the weights on the similarity features (whether the clustering should be more based on the rule contents of measures). The rules can be exported into a text file as a human-readable text or in JSON format. Since the mining algorithm and the RuleSet structure do not work with IRIs and literals directly but with their IDs assigned during the creation of indices, when the rule set is cached into a file, the appropriate instance of Index is needed to restore the rule set, so that the IDs from which the rules consist can be mapped to their IRIs and literals (the rules can be *resolved*).

```
Dataset("yago.tsv")
    .filter(!_.triple.predicate.hasSameUriAs("participatedIn"))
    .discretize(DiscretizationTask.Equifrequency(3))
    (_.triple.predicate.hasSameUriAs("hasNumberOfPeople"))
    .mine(Amie()
    .addConstraint(RuleConstraint.WithInstances(true))
    .addPattern(AtomPattern(predicate = Uri("hasNumberOfPeople")) =>: None)
    .addPattern(AtomPattern(predicate = Uri("hasNumberOfPeople"))))
    .computePcaConfidence(0.5)
    .sorted
    .export("rules.json")
```

Listing 5.1: An example of a rule mining workflow with RDFRules Scala API (Source: [25])

[7] https://github.com/KIZI/EasyMiner-Discretization

6. Leveraging a Combination of OLAP Cubes and Knowledge Graphs

OLAP analysis and graphical visualisation suggest itself as natural ways of analyzing statistical linked data. Section 2.3 describes how the statistical data can be represented as RDF using the Data Cube Vocabulary. The effort of anylyzing the statistical RDF data is focused on making use of the *open* rather than *linked* nature of Linked Open Data. Available RDF data is extracted as is and either loaded into an OLAP system and treated as any other OLAP data [13] or it is analyzed by SPARQL queries.[20]

In [12] the application of GUHA [18] association rules for mining over the aggregated data was suggested as a complement to the standard OLAP analysis. The aggregated observations are treated as individual transactions in tabular data. The mined rules describe strong relations in the cube and can serve as a guide unusual trends that would be otherwise hard to find by manually browsing the OLAP cube.

The following text builds on the idea that some trends in OLAP cube can be expressed in as association rules and elaborates the possibilities of mining association rules by the AMIE algorithm over aggregated data in a form of a data cube. The AMIE algorithm mines rules directly over RDF data which can therefore be enriched by another relations describing the cube's dimension values extractable from other LOD sources so that the *linked* nature of LOD is exploited as well.

6.1 Mining from RDF representation of Data Cube

We are going to aim to mine association rules that describe how a certain measure's values are determined by values of one or more dimensions meaning that the observations of the examined cube corresponding to fixed values of some dimensions (slicing and dicing) tend to contain a measure values with a certain characteristic. The antecedent of the rule would basicaly specify a certain slice (a subset of observations) of the examined cube in which the measured values deviate from the whole and the consequent of the rule would describe the nature of this deviation.

6.1.1 Basic Shape of the Rules

Such a rule (the predicted characteristic of the subset of observations) has to be deducible by the algorithm so its head atom has to be *instantiatable* by triples contained in the examined data set so the head's predicate will be one of the cube's measures, the head's subject will be a variable representing observations for which the rule's antecedent is valid and the head's object

will be a specific constant of a discrete measure's value, that is predicted to be associated with all the observations that satisfy the antecedent. For example, imagine a very small data cube which arose from a survey on the life satisfaction among men and women in several european countries. In each country a male and a female respondents were asked to rank their life satisfaction either as *low*, *moderate* or *high*. So the resulting data cube contains two dimensions of *sex* and *country* and a single measure of life satisfaction. Each observation in the cube below corresponds to the most common answer in the survey for the particular country and sex.

	France	Germany	Poland	Slovakia	Austria
Men	moderate	high	moderate	high	high
Women	low	moderate	high	low	low

Table 6.1: Basic life satisfaction survey

Based on the results of the survey in the table above we could determine that men across the countries tend to have a high life satisfaction. This fact can be expressed as an association rule that is deducible by the AMIE algorithm.

$$(?o\ sex\ Male) \Rightarrow (?o\ lifeSatisfaction\ High) \tag{6.1}$$

We can calculate measures of significance of this rule from the values in the table. Let's calculate the support for now. Since the object in the rule's head is a constant, the support of the rule corresponds only to the distinct observations that belong to men and have the correct measure value, i.e. 3.

6.1.2 Measures in the Body

There are two possible ways to structure a data cube containing more than two measures. Either each observation is assigned multiple measures (let's denote it as `1xN`) or there is one observation for each measure and the context of the observation (let's denote it as `Nx1`). If the cube has more than one measure, one of the measures can be used in the rule's body to further specify the subset of the observations. Let's extend the survey by another question. The respondents also had to rank their salary as either *low*, *decent* or *high*.

		France	Germany	Poland	Slovakia	Austria
lifeExpectation	Men	moderate	high	moderate	high	high
	Women	low	moderate	high	low	low
salary	Men	low	decent	low	decent	decent
	Women	low	high	decent	low	decent

Table 6.2: Two measures in the cube

The table 6.2 corresponds to the `Nx1` style of structuring the cube's measures. Now if the rule's body can contain a new atom demanding a certain value of the new measure for the observations:

$$(?o\ sex\ Male) \wedge (?o\ salary\ Decent) \Rightarrow (?o\ lifeSatisfaction\ High) \tag{6.2}$$

6.1.3 Measures of Confidence

More interesting is an expression of a prediction's quality. That lead us to the measures of confidence. The standard confidence of a rule is calculated as the ratio of observations satisfying the rule's body and the rule's head over the number of observations satisfying the rule's body in the cube.

Let's calculate the standard confidence of the rule 6.1 when mined from both tables. For the table 6.1 it would be $3 \div 5 = 0.\overline{6}$. For the table 6.2 it would be $3 \div 10 = 0.\overline{3}$. Even though the table 6.2 contains the same data for the life satisfaction, it gives a lower standard confidence. It is because of the observations of the salary measure that are assigned the male sex. They also enter the denominator of the standard confidence.

The PCA confidence would not consider the observations of average salary the counterexamples, so it would not suffer from this distortion. If each observation is assigned all measures in the table (`1xN`), the standard confidence and PCA confidence values would be identical for each rule.

6.1.4 Commensurability

In the previous examples, the measured values were discrete but data cubes usually contain continuous numerical values on which aggregation operations (sum, average, etc.) are possible. In order to ensure, that the rules can achieve a reasonable support, the numerical values have to be discretized (see 4.3.1) and replaced by intervals. But one cannot simply discretize all measure's values at once. The dimension values can be structured into a hierarchy so the measures belonging to dimension values of different levels in the hierarchy are not comparable.[12] Irrespective to a chosen discretization approach, it is inadmissible to discretize values belonging to different disproportionate contexts.

One way (style H) to solve this is to dice the cube having disproportionate dimension values into a set of smaller subcubes, in which the dimension values belong to the same level of a concept hierarchy. Measured values can be then discretized into intervals in each subcube separately. Number of subcubes that the main cube has to be divided into depends on the number dimensions and the number of levels in each dimension's hierarchies. The rules would be mined either separately on each subcube or for each observation a triple with the

assignment of the observation to its subcube would have to be added into the data set and this assignment would have to be stated in the body of the rules.

An alternative (style C) to this proposed in [14] is to cluster the values of each measure in the whole cube by a clustering algorithm a then to create intervals inside those clusters. This would be useful when there is a significant difference in measurements belonging to the same level of hierarchy for a dimension, or when the hierarchy of the dimension values is not known. [14] gives an example of sales distribution among different products in a hypermarket. The number of sales of bakery will be incomparable to number of sales of electric razors. In the context of the AMIE algorithm, the assignment of each observation to each measure's cluster would have to be added to the data set and then expressed in the rule's body, otherwise also the observations belonging to different clusters would be considered counterexamples and the confidence would not be computed properly.

One disadvatage of this approach is that there is no clear interpretation of the generated subcubes. Their observations can belong to different levels of the concept hierarchy in a dimension. Unlike with the previous approach where a generated subcube could be described as for example *population in districts by age category*.

The following example explains the style H. Adam and Beatrice work as food delivery persons. Data about the amount of their delivered orders in 2020 is entered into a data cube with two dimensions of the person delivering and the time interval to which the number of delivered orders relates. The length of time intervals varies. They are either whole weeks or whole months.

	1st Week	2nd Week	3rd Week	January	February	March
Adam	50	55	40	150	250	200
Beatrice	30	40	20	100	120	130

Table 6.3: Numbers of Adam's and Beatrice's delivered orders

The values belonging to weeks are not comparable with the values belonging to months. The cube will be sliced into two cubes, with one containing the observations of week and the second with observations of months. Values will be then discretized in both cubes separately. For the sake of simplicity, the values will be discretized into two equifrequent intervals.

	1st Week	2nd Week	3rd Week	January	February	March
Adam	ef2/2_1	ef2/2_1	ef1/2_1	ef2/2_2	ef2/2_2	ef2/2_2
Beatrice	ef1/2_1	ef1/2_1	ef1/2_1	ef1/2_2	ef1/2_2	ef1/2_2

Table 6.4: Discretized numbers of Adam's and Beatrice's delivered orders

Let's have a rule 6.3 stating that Beatrice's weekly delivered orders *are low* (in the lower of the two equifrequent intervals). The assignment of the valid observations to the correct subcube has to be part of the body so that the Beatrice's montly order values are not considered counterexamples. In Data Cube Vocabulary, this assignment is provided by the $qb : dataSet$

property.

$$(?o\ person\ Beatrice) \wedge (?o\ qb : dataSet\ Week) \Rightarrow (?o\ orders\ ef1/2\_1) \tag{6.3}$$

Structure Style A

The question is to how many intervals the measures should be discretized into. If creating too many intervals, more specific rules should be found but they happen to have a lower support. When performing the equifrequent discretization, coarser rules are found for lower number of intervals. To avoid guessing which discretization parameters suit best the data, multiple discretizations can be performed with different parameters for various discretization algorithms.

As it was already mentioned, the preprocessed cube should to be cut into subcubes with commensurable observations, measures in these subcubes have to be discretized separately and only after that the triples can be merged and performed the mining tasks on.

```
@prefix qb: <http://purl.org/linked-data/cube#> .

<o1> qb:dataSet <original-dataset> ;
    <dimension1> <dimension1value1> ; <dimension2> <dimension2value1> ;
    <measure1> 25000 ;
    <measure2> 3 .

<o2> qb:dataSet <original-dataset> ;
    <dimension1> <dimension1value2> ; <dimension2> <dimension2value2> ;
    <measure1> 10000 ;
    <measure2> 10 .
```

Listing 6.1: Observations example

That means that the number of overall *measurements* multiplies by the number of distinct discretizations. A situation has to be avoided, when the instantiations of variable representing observations are involving observations not only from various subcubes but also those observations that are assigned measurements from dijunct sets of intervals (distinct discretizations). The result of a approach (let's call it style A) avoiding this problem on the sample data in 6.1 is shown in the listing 6.2.

Each application of a discretization algorithm will create a new measurement triple for each measure and observation with an object of the assigned interval based on the discretization algorithm and the parameter. The objects in triples assigning the observations to the whole data set will be changed to point to the particular subcube. In the example 6.2 two discretizations for each measure were performed on the two observations. The same pair of discretizations was performed on the two distinct measures, but that does not have to be so. Assigning multiple triples of the same measure is fine as far as the measure is differently discretized.

```
@prefix qb: <http://purl.org/linked-data/cube#> .

<o1> qb:dataSet <subcube1> ;
    <dimension1> <dimension1value1> ; <dimension2> <dimension2value1> ;
    <measure1> <subcube1_ef3_measure1_3>, <subcube1_ef10_measure1_2> ;
    <measure2> <subcube1_ef3_measure2_2>, <subcube1_ef10_measure2_1> .

<o2> qb:dataSet <subcube2> ;
    <dimension1> <dimension1value2> ; <dimension2> <dimension2value2> ;
    <measure1> <subcube2_ef3_measure1_3> , <subcube2_ef10_measure1_2> ;
    <measure2> <subcube2_ef3_measure2_1> ,<subcube2_ef10_measure2_1> .
```

Listing 6.2: Structure style A

Structure Style B

A different way of structuring the discretized measures (let's call it style B) within the observations would remove both distortions while also permitting to mine over the whole original cube. Each performed discretization would introduce a new derived measure whose name could be chosen as the name of the original measure suffixed by type of discretization with its parameters and the observation's subcube.

```
@prefix qb: <http://purl.org/linked-data/cube#> .

<o1> qb:dataSet <subcube1> ;
    <dimension1> <dimension1value1> ; <dimension2> <dimension2value1> ;
    <subcube1_ef3_measure1> <subcube1_ef3_measure1_3> ;
    <subcube1_ef10_measure1> <subcube1_ef10_measure1_2> ;
    <subcube1_ef3_measure2> <subcube1_ef3_measure2_2> ;
    <subcube1_ef10_measure2> <subcube1_ef10_measure2_1> .

<o2> qb:dataSet <subcube2> ;
    <dimension1> <dimension1value2> ; <dimension2> <dimension2value2> ;
    <subcube2_ef3_measure1> <subcube2_ef3_measure1_3> ;
    <subcube2_ef10_measure1> <subcube2_ef10_measure1_2> ;
    <subcube2_ef3_measure2> <subcube2_ef3_measure2_1> ;
    <subcube2_ef10_measure2> <subcube2_ef10_measure2_1> .
```

Listing 6.3: Discretization style B

The style B has one side effect and that is that the new derived measures become correlated, e.g. when a measure value belongs to the lowest of 10 intervals then it surely belongs to the lowest of 3 intervals. That would make it hard to mine rules which specify the subset of observations also by one of the measures of the observations, because a rule pattern that allows or even enforces an atom with a measure predicate in the rule's would be matched with those correlations.

6.1.5 Mining the Whole Cube vs. Each Subcube Separately

The discretized subcubes can either be performed mining tasks on separately or they could remain as a singe data set. The second option is less demanding. A single index is created from which a single rule set is generated. It distorts the lift measure though, because it increases the denominator of the head confidence.

To show the effect let's return to the example cube 6.4 of Adam and Beatrice and let's compute the head confidence[1] of the rule first for the case when the subcubes[2] are mined on seperately. The examined rules would be contained in the rule set generated from mining the *weekly* subcube.

$$hconf(\vec{B} \Rightarrow H) = \frac{\#s : \exists\langle s,p,o\rangle \prec (?a\ p\ C)}{\#s : \exists\langle s,p,o\rangle \prec (?a\ p\ ?b)} = \frac{3}{6} = \frac{1}{2}$$

If we mined over the whole cube, the head confidence's denominator would consider all observations in the cube.

$$hconf(\vec{B} \Rightarrow H) = \frac{\#s : \exists\langle s,p,o\rangle \prec (?a\ p\ C)}{\#s : \exists\langle s,p,o\rangle \prec (?a\ p\ ?b)} = \frac{3}{12} = \frac{1}{4}$$

Also in each rule it would have to be ensured, that the set of observations is limited to a certain subcube. For each cube in a rule the body of a rule has to contain an atom of a pattern $(?o\ qb : dataSet\ AnyConstant)$.

However, if the cube 6.4 were structured by the style B, the lift would not be distorted because the predicate of a rule generated from such data would appear only in the correct subcube, so only the right observations would be considered.

6.1.6 Note to Head Coverage

The head coverage is calculated as support divided by the number of distinct triples with the head's predicate, so-called head size.

The head coverage is distorted both by

1. mining over the whole cube at once
2. by performing multiple discretizations of the same measure while using the structure style A

When using the structure style A, each performed discretization over a measure increases the head size by the number of observations. So does the inclusion of triples with observations not

[1] We calculate only the head confidence as the denominator of the lift formula since the lift formula's numerator (confidence) does not change.

[2] That means one subcube of the weekly order amounts and one subcube of the monthly order amounts.

compatible with the rule regardless of the structure style used. To show this, let's compute the head size of the rule 6.3 when at first mined only from the *weekly* subcube.

$$hsize(\vec{B} \Rightarrow (s\ p\ o)) = \#\langle s,p,o\rangle \in KG : \langle s,p,o\rangle \prec (?a\ p\ ?b) = 6$$

There are only 6 triples with the head's predicate in the whole mined data set. Now let's perform a second discretization over Adam's and Beatrice's data cube and also mine over the whole cube.

	1st Week	2nd Week	3rd Week	January	February	March
Adam	ef2/2_1	ef2/2_1	ef1/2_1	ef2/2_2	ef2/2_2	ef2/2_2
	ef3/3_1	ef3/3_1	ef2/3_1	ef2/3_2	ef3/3_2	ef3/3_2
Beatrice	ef1/2_1	ef1/2_1	ef1/2_1	ef1/2_2	ef1/2_2	ef1/2_2
	ef1/3_1	ef2/3_1	ef1/3_1	ef1/3_2	ef1/3_2	ef2/3_2

Table 6.5: Data cube discretized multiple times

Now the head size will be higher because all 12 observations are considered and the head size will come up as 12 multiplied by the number of performed discretizations.

$$hsize(\vec{B} \Rightarrow (s\ p\ o)) = \#\langle s,p,o\rangle \in KG : \langle s,p,o\rangle \prec (?a\ p\ ?b) = 12 * 2 = 24$$

When using the structure style B, the measure URI for each performed discretization is different so the head coverage of a rule generated from such data can be interpreted as the ratio of the rule's support and the number of appropriate observations regardless of whether the whole cube is mined at once or the subcubes are mined separately because the predicate of the rule will be anchored both to the appropriate subcube and to a single performed discretization.

Just as with the support, that would still disadvantage finer discretizations because they would tend to have lower and support while the head size would be constant across all rules with associated with the same subcube and the same original measure.

The support of a rule is also affected by the number of observations of the subcube to which the rule is anchored. The rules generated for a bigger subcube will tend to have a higher support compared to rules from a smaller subcubes because there will be more observations that can meet the condition of the rule's body. But this effect is cancelled out in the head coverage because with more observations its numerator (support) grows but its denominator grows as well.

So for a cube constisting of a high number of subcubes it can be suggested to use the style B of expressing the discretized measures and to mine the cube at once with a defined head coverage parameter. That is because the distortion of both lift and head coverage caused by mining the whole is cancelled out by the usage of the structure style B

A low number of subcubes could be mined separately with a custom minimum support threshold defined for each subcube to tweak each subcube's rule set. And since the separate mining of each subcube solves the lift distortion by itself and a minimum support threshold would be used instead of a minimum head coverage, the measure expression style *A* could be used because it's propably easier to perform.

	style A	style B
mining the whole cube		high no. of subcubes
mining subcubes separately	manageable no. of subcubes	

Table 6.6: Pre-processing suggestions based on the nature of the examined cube

6.2 Appending RDF Data to the Data Cubes

So far the showed rules specified the subset of observations only by exactly one value for a dimension, just as e.g. the Apriori algorithm's rules contain only one category for each atribute. Slicing the subset of observations just by one value per dimension would yield just a contrained set of rules. The language bias of the AMIE algorithm does not allow a collection (disjunction) of values in the atom's object. It can either be a constant or a variable. But a variable can appear in another atom in which it is attributed a *property*. That way the variable acts as a collection of entities sharing this property.

If the dimension values represent real-world entities such as geographical areas, persons, organizations, etc., such properties can be found in the form of RDF triples published in public knowledge graphs. Those triples can enter the algorithm together with the triples describing the data cube. Let's continue with the example of Adam and Beatrice. Table 6.7 shows a simple data cube containing the numbers of their daily delivered orders from July 14th to 17th. Each cells contains the original value of daily orders plus its interval after the equifrequent discretization into four intervals.

Orders	07-14	07-15	07-16	07-17
Adam	10	7	6	0
	ef4/4	ef3/4	ef2/4	ef1/4
Beatrice	9	0	8	4
	ef4/4	ef1/4	ef3/4	ef2/4

Table 6.7: Data cube of daily delivered orders

Both also publish their personal information as RDF with the *FOAF* Vocabulary[3] on their blogs. The listing 6.4 shows a deluge of both person's information.

If this data is merged with the data cube, one of the rules found by AMIE algorithm would be a rule stating that a when a person has birth day, their number of delivered orders in

[3]http://www.foaf-project.org/

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<http://adamsmith.xyz/#me>
    a foaf:Person ;
    foaf:name "Adam Smith" ;
    foaf:givenname "Adam" ;
    foaf:family_name "Smith" ;
    foaf:birthday "07-17"^^xsd:string ;
    foaf:homepage <http://www.adamsmith.xyz> .

<http://beatricet.com/#me>
    a foaf:Person ;
    foaf:name "Beatrice Taylor" ;
    foaf:givenname "Beatrice" ;
    foaf:family_name "Taylor" ;
    foaf:birthday "07-15"^^xsd:string ;
    foaf:homepage <http://www.beatricet.com> .
```

Listing 6.4: RDF data published on Adam's and Beatrice's personal blogs

that day is in the lowest of four equifrequent intervals. For this simple example it could also predict the exact zero. Probably because when someone has birth day, they throw a party with their friends or with family and they do not bother with delivering meals.

$$(?o\ day\ ?b) \wedge (?p\ foaf:birthday\ ?b)$$
$$\wedge (?o\ person\ ?p) \wedge (?o\ qb:dataSet\ Day) \Rightarrow (?o\ orders\ ef1/4)$$

This example is simplified in the way that it assumes that the IRIs representing Adam and Beatrice in the data cube are identical to the IRIs they assigned to themselves and also that the data cube uses the same date format as the *FOAF* Vocabulary suggests. Which would not because the cube's dates represent actual days, whereas birth day is just a combination of month and day repeating every year, so this relation would have to be inferred by another triples whose atom would have to be in the rule. The dimension values do not just have to be *described* directly. The rule can contain a *chain* of atoms not instantiated by the cube's triples starting with the dimension value's variable.

Real-world entities or concepts can be assigned multiple identifiers in the LOD environment. In order to the AMIE algorithm to make the right connection between the dimension values and triples from other sources describing them, the identifiers have to either be unified or triples stating their equivalence e.g. with the *owl:sameAs* predicate have to be added into the data set and those connections have to be part of the rules.

6.3 Finding Rules Concerning Multiple Cubes

The same dimension values of a data cube can be present in other data cubes either with the same or a different identifier. For example both a cube containing procurement statistics of a

country and a cube containing demography statistics of the country would have a dimension of the reference areas in the country. The measures attributed to a dimension value in a data cube can be considered properties describing the dimension value, therefore there is a possibility to use it to specify the subset of observations in an examined cube just as the RDF data from public knowledge graphs. But one has to remember that the measures are given in the context of all dimensions in the cube.

Let's have a first example of two small data cubes. One contains averages salary statistics by region, year and sex. The second cube contains average pension statistics by region and year. The cubes vary in the number of dimensions but they share the dimensions of region and year. Do not mind that some cells are stained for now.

Dimensions		Salaries			Pensions		
		2010	2011	2012	2010	2011	2012
Region 1	Male	High	High	High	Medium	High	High
	Female	Medium	Medium	High			
Region 2	Male	Low	Medium	High	Low	Low	Low
	Female	Low	Low	Low			

Table 6.8: Two simple cubes with some shared dimensions

We can imagine this rule generated by AMIE algorithm:

$$(?o1\ qb : dataSet\ Salaries) \wedge (?o1\ salary\ High) \wedge (?o1\ sex\ Male)$$
$$\wedge (?o2\ region\ ?r) \wedge (?o1\ region\ ?r) \wedge (?o1\ year\ ?y)$$
$$\wedge (?o2\ year\ ?y) \wedge (?o2\ qb : dataSet\ Pensions) \Rightarrow (?o2\ pension\ High)$$

The rule states that if there is an observation in the *Salaries* cube assigned to males and to some region and year and this observation's measure for average salary reads *High*, then all observations from the *Pensions* cube assigned to the same region and year have the average pension measure value of *High*. For each observation from the *Salaries* cube that satisfies the rule's body there is exactly one observation from the *Pensions* cube that does as well.

This is illustrated by the red colored cell in the *Salaries* cube and the red colored cell in the *Pensions* cube. The observation corresponding to year 2011 and Region 1 and males in the *Salaries* cube reads average salary value *High*. And the rule states that if such observation exists, all the observations in the *Pensions* cube corresponding to the same year and region have the *High* value of average pension. The only observation in the *Pensions* cube for the year 2011 and Region 1 is the very red colored observation.

The rule predicted the right value in this case. But it would also predict the *High* value for the observation corresponding to year 2010 and Region 1, because in the *Salaries* cube there exists a observation for Region 1, year 2010 and males, that has the *High* value of average salary. But the value of the observation in the *Salaries* cube that corresponds to Region 1

and year 2010 has the *Medium* value of the average pension. So the prediction of the rule was wrong in this case.

All observations of the *Pensions* cube, that satisfy the rule's body, correspond to the subset of observations, for which the rule predicts the *High* vale of average pension. Table 6.9 shows the span of the subset specified by the rule.

Dimensions		Salaries			Pensions		
		2010	2011	2012	2010	2011	2012
Region 1	Male	High	High	High	Medium	High	High
	Female	Medium	Medium	High			
Region 2	Male	Low	Medium	High	Low	Low	Low
	Female	Low	Low	Low			

Table 6.9: Illustration of the specified subset

PCA confidence of the rule would be $2 \div 4 = 0.5$ as 2 out of 4 observations satisfying the rule's body also satisfy the rule's head. The lift of the rule would be $(2 \div 4) \div (2 \div 6) = 1.5$ as 2 out of 6 observations in the *Pensions* cube have the *High* value of average pension.

The interpretation of the rule: *If there is a high average salary for males in a region in a year, there is a high average pension in that region in that year.*

6.3.1 Open Dimensions

Let's refer to the cubes whose observations are instantiated in the variable that appears in a rule's head as *head cubes* for the rule and the cubes whose observations are instantiated in the variable that appears only in a rule's body as *body cubes* for the rule. There can be one or more body cubes and exactly one head cube in a rule. The *Salaries* cube in the rule above is a body cube and the *Pensions* cube is a head cube.

If a cube's dimension appears in the rule, let's refer to the dimension as *closed* for the rule. And if it does not appear in the rule, let's refer to the dimension as *open* for the rule. All the dimensions of the *Salaries* cube are closed for the rule and so are all the dimensions of the *Pensions* cube.

If one or more dimensions of a body cube are open in a rule, then the interpretation of the rule becomes complicated. The rule below was mined from the same cubes with the same structure, but the *Salaries* cube has one open dimension (sex) in the rule.

$$(?o1\ qb : dataSet\ Salaries) \wedge (?o1\ salary\ High) \wedge (?o2\ region\ ?r)$$
$$\wedge (?o1\ region\ ?r) \wedge (?o1\ year\ ?y) \wedge (?o2\ year\ ?y)$$
$$\wedge (?o2\ qb : dataSet\ Pensions) \Rightarrow (?o2\ pension\ High)$$

For each observation in the *Pensions* cube that satisfies the rule's body, there are two observations corresponding to the same year and region in the *Salaries* cube that do as well. That is illustrated by the green colored cells in table 6.8. The rule states that if there exists an observation for a region and a year in the *Salaries* cube whose value of average salary reads *High*, then all observations in the *Pensions* cube corresponding to the same region and year will have the *High* value of average pension. The observation with the correct value in the *Salaries* cube could be **either** for males **or** for females.

So the interpretation of the rule is: *If **either** men **or** women have high salary in a region in a year then people have high pensions in the region in that year.*

Following example shows how the interpretability of a rule is affected when all body cubes have only closed dimensions but one or more head cube's dimensions is open. The cubes have now a different structure.

Dimensions		Salaries			Pensions		
		2010	2011	2012	2010	2011	2012
Region 1	Male	Medium	High	High	Medium	High	High
	Female				Medium	Medium	Medium
Region 2	Male	Low	Medium	High	Low	Low	Low
	Female				Medium	Low	Low

Table 6.10: Different structure of the cubes

The *Salaries* cube has the dimensions of region and year and the *Pensions* cube has the dimensions of region, year and sex. In the rule below the head cube has one open dimension of sex.

$$(?o1\ qb : dataSet\ Salaries) \wedge (?o1\ salary\ High) \wedge (?o2\ region\ ?r)$$

$$\wedge (?o1\ region\ ?r) \wedge (?o1\ year\ ?y) \wedge (?o2\ year\ ?y)$$

$$\wedge (?o2\ qb : dataSet\ Pensions) \Rightarrow (?o2\ pension\ High)$$

For each observation in the *Salaries* cube satisfying the rule's body there are 2 observations in the *Pensions* cube assigned to the same region and year. This is illustrated by the orange cells in the table 6.10. The rule states that if there exists an observation in the *Salaries* cube for a year and a region that has the *High* value of the average salary, then all observations in the *Pensions* cube assigned to the same region and year will have the *High* value of average salary. The rule would predict the value for both male **and** female observations in the *Pensions* cube.

So the interpretation of the rule is: *If there is a high salary in a region in a year then both men **and** women have high pensions in the region in that year.*

6.3.2 Summary

The interpretation of a rule that describes a relation between multiple data cubes depends on the structure (their dimensions and range of those dimensions) of those cubes. If there is an open dimension of the head cube, the number of the added *and* statements corresponds to the product of all distinct values of open dimensions of the head cube. If there is an open dimension of a body cube, the number of added *or* statements corresponds to product of all distinct values of open dimensions of all open body cubes. Therefore open dimensions in the body cubes should be avoided unless the number of distinct values of an open dimension is very low. The structures of the cubes does not necessarily need to be identical. The body cubes function just as any other way to specify the subset of observations.

6.4 Note to the Rule Atoms Order

The rules concerning multiple cubes would be much more readable if the atoms in the rule's body belonging to the same cube were put abreast. For example the last rule could be rewritten as:

$$(?o1\ qb : dataSet\ Salaries) \wedge (?o1\ salary\ High) \wedge (?o1\ region\ ?r)$$

$$\wedge (?o1\ year\ ?y) \wedge (?o2\ qb : dataSet\ Pensions) \wedge (?o2\ region\ ?r)$$

$$\wedge (?o2\ year\ ?y) \Rightarrow (?o2\ pension\ High)$$

However, such rule would never be generated by the AMIE algorithm. Problem is with the $(?o2\ region\ ?r)$ atom, which is body's second atom from the left. It introduces a new variable $?r$. No rule refinement operator would append this atom to the intermediate rule $(?o2\ year\ ?y) \Rightarrow (?o2\ pension\ High)$. The dangling atom operator would append atom with a new variable and the open variable $?y$, thus closing this variable. The closed atom operator would add an atom with the variables $?o2$ and $?y$, thus closing the variable $?y$ as well. The algorithm cannot add a new variable to a rule without closing an open variable in the rule. One has to bear in mind how the algorithm traverses the rules search space and how it applies the refinement operators when e.g. defining the rule patterns the algorithm is allowed to mined, which the RDFRules framework allows.

A person using the framework could tend to write a rule pattern and understand a rule from left to right, from the body to the head, but the algorithm creates the rules in the opposite direction. That also implies that the atom anchoring the observation variable to a specific cube with the $qb : dataSet$ should be the rightmost (and therefore the first to be appended to the rule) of the atoms containing the cube's observation variable, so that the possible instantiations of the variable are restricted as soon as possible.

7. Experiment

This section describes an experiment of mining associtation rules from RDF data compiled of statistical data structured by the Data Cube Vocabulary and facts pulled from the Wikidata data set that was performed as an practical part of this work. The statistical data come from two sources. The first one is the Czech Social Security Administration and the second one is the Czech Statistical Office. Analysis was performed using the Scala API of the reference implementation of the RDFRules algorithm. The following sections describe how the available data had to be preprocced to give reasonable results in combination with KG data. The preprocessing was perfomed partly by the implementation's API itself, partly by performing SPARQL queries over the data. The described method can be taken inspiration from when performing similar analysis ie. association rule mining task over the multidimensional data merged with loosely structured graph data.

7.1 Czech Social Security Administration's Data Cubes

Czech Social Security Administration (CSSA) is a czech public administration organtisation responsible for collecting social security premiums and contributions to the state employment policy. Since 2015 the organization publishes its statistical yearbook datasets and other (vocabularies, code lists and datasets containing data concerning the internal operation of the organization) in the form LOD and became one of the first czech public institutions to do so. The yearbook statistical data sets are modeled using Data Cube Vocabulary. Their dimension values are represented by the SKOS vocabulary. The organization has published 73 datasets so far. All these datasets are downloadable as dumps[1] or accesible througth a SPARQL endpoint[2]. The CSSA's URIs are dereferenceable.

The largest of the data cubes published is `cssa-d:duchodci-v-cr-krajich-okresech`[3]. From now on it will be denoted as *Pensions*. It contains 368 118 observations spread over four dimensions: reference area[4], reference period[5], sex[6] and pension kind[7]. Observations are assigned three measures: the average amount of pension[8], the average age[9] and the number of persons[10]. Each observation is assigned only one measure.

In this particular data set there are 102 distinct values of the dimension of reference area:

[1] http://data.cssz.cz/web/otevrena-data/katalog-otevrenych-dat
[2] http://data.cssz.cz/web/otevrena-data/sparql-query-editor/
[3] https://data.cssz.cz/resource/dataset/duchodci-v-cr-krajich-okresech
[4] https://data.cssz.cz/ontology/dimension/refArea
[5] https://data.cssz.cz/ontology/dimension/refPeriod
[6] https://data.cssz.cz/ontology/dimension/pohlavi
[7] https://data.cssz.cz/ontology/dimension/druh-duchodu
[8] https://data.cssz.cz/ontology/measure/prumerna-vyse-duchodu-v-kc
[9] https://data.cssz.cz/ontology/measure/prumerny-vek
[10] https://data.cssz.cz/ontology/measure/pocet-duchodcu

```
@prefix qb: <http://purl.org/linked-data/cube#> .
@prefix cssa-om: <https://data.cssz.cz/ontology/measure/> .
@prefix cssa-d: <https://data.cssz.cz/resource/dataset/> .
@prefix cssa-od: <https://data.cssz.cz/ontology/dimension/> .

<https://data.cssz.cz/resource/observation/duchodci-v-cr-krajich-okresech/2017-12-31/
     prumerna-vyse-duchodu-v-kc/pk_srnvm/vc.35/m>
    a qb:Observation ;
    qb:dataSet cssa-d:duchodci-v-cr-krajich-okresech .
    qb:measureType cssa-om:prumerna-vyse-duchodu-v-kc ;
    cssa-od:druh-duchodu <https://data.cssz.cz/resource/pension-kind/PK_SRNVM_2010> ;
    cssa-od:pohlavi <https://data.cssz.cz/ontology/sdmx/code/sex-M> ;
    cssa-od:refArea <https://data.cssz.cz/resource/ruian/vusc/35>;
    cssa-od:refPeriod <https://data.cssz.cz/resource/reference.data.gov.uk/id/gregorian-day/2017-12-31>;
    cssa-om:prumerna-vyse-duchodu-v-kc 6622.0;
```

Listing 7.1: Example of an observation from *pensions* data set

14 regions (NUTS 3 administrative units, czech translation in singular nominative is *kraj*) including Prague, 77 districts (*okres*) also including Prague, 10 Prague districts (*správní obvod*) and a value representing the state in total. Each entity representing a reference area is assigned an unique numerical identifier which corresponds to this area's identifier in the official Registry of Territorial Identification, Addresses and Real Estate (RTIAR) runned by the State Administration of Land Surveying and Cadastre (SALSC). RTIAR codes are reference codes by law, so it is obligatory for CSSA to use them and have them correct.

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .

<https://data.cssz.cz/resource/ruian/vusc/35>
    a <https://data.cssz.cz/ontology/ruian/Vusc> , skos:Concept ;
    <http://www.w3.org/2002/07/owl#sameAs> <https://linked.cuzk.cz/resource/ruian/vusc/35> ;
    skos:inScheme <https://data.cssz.cz/resource/ruian/ConceptScheme> ;
    skos:notation "VC.35" ;
    skos:prefLabel "Jihočeský kraj" .
```

Listing 7.2: Dereferenced proxy entity of the South Bohemian Region

There are official URIs of this registry but CSSA datasets do not use them directly. The entities for the dimension of reference area and other dimensions in the dataset *Pensions* and all other data sets of CSSA with the dimension of reference area work as so-called *proxy entities*. This means that instead of using the original code list item URIs directly as objects in the RDF triples, it uses their equivalents defined in the internal code lists. These equivalents are connected to the original URI by the `owl:sameAs` statement. These proxies can then contain data specific to CSSA e.g. labels. Another advantage of this is, that these URIs are dereferenceable to the CSSA domain and their versioning is under the control of CSSA and they can be easily redirect to a different equivalent code list. Previously the proxy entities of the reference area were directed to the unofficial transformation of the Opendata.cz initiative[11].

Another dimension whose values work as proxy entities is the dimension of reference period.

[11] https://linked.opendata.cz/dataset/cz-ruian

This dimension divides the observations into one year intervals. This applies to all other CSSA data cubes containg the reference period dimension. The data set vary in the overall covered period. The first covered time period is the year 2001 and the last covered year of all data sets is the year 2019. The entities link to the `data.gov.uk` Time Intervals[12] OWL ontology. Usage of this entities is, however, not unified. In some data sets intervals are assigned an entity representing a year, in other they are assigned an entity representing the last day of the corresponding year. All of them are, however, representing a period of a whole year.

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .

<https://data.cssz.cz/resource/reference.data.gov.uk/id/gregorian-year/2017>
    a skos:Concept ;
    <http://www.w3.org/2002/07/owl#sameAs> <http://reference.data.gov.uk/id/gregorian-year/2017> ;
    skos:inScheme <https://data.cssz.cz/ontology/years/YearsScheme> ;
    skos:notation "2017" ;
    skos:prefLabel "2017" .
```

Listing 7.3: Dereferenced proxy entity of the year 2017

The *Pensions* dataset uses two distinct schemes for the pension kinds, because in 2010, the official categorization of pensions was changed in the czech legislation. Only the observations assigned to year 2008 are divided according to the old pension scheme. All other year's observations correnspond to the new pension scheme. So one can not simply multiply the numbers of distinct values for each dimension and the number of measures to get the total number of observations for this particular cube. The URIs of both pension kind schemes are suffixed either by `_2008` (31 of them) for the old scheme or `_2010` (37 of them) for the new scheme. Not all entities correspond to a particular kind of pension. Some of them represent an aggregation over related pension kinds or simply an aggregation over all of the pension kinds.

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .

<https://data.cssz.cz/resource/pension-kind/PK_SRNVM_2010>
    a <https://data.cssz.cz/ontology/pension-kinds/PensionKind> , skos:Concept ;
    skos:altLabel "SRNVM"@cs ;
    skos:exactMatch <https://data.cssz.cz/resource/pension-kind/PK_SRNVM> ;
    skos:inScheme <https://data.cssz.cz/ontology/pension-kinds/PensionKindScheme_2010> ;
    skos:notation "PK_SRNVM" ;
    skos:prefLabel "Starobní důchod SRN vyplácený v souběhu s vdoveckým důchodem"@cs .
```

Listing 7.4: Dereferenced pension kind

The dimension of sex consists of three distinct values: dimension of male pension, dimension of female pensions and a value for both sexes. The values are proxy entities as well linking to the SDMX representations of sexes[13]

[12] http://old.datahub.io/dataset/data-gov-uk-time-intervals
[13] http://purl.org/linked-data/sdmx/2009/code

7.2 Czech Statistical Office's Data Cubes

The Czech Statistical Office (CZSO) is the main public organization responsible for collecting and analyzing statistical data in the Czech Republic. This organization is for example responsible for the state's census. Data about demography, economics, education, health care, etc. are made availabe on the organization's website[14] in a form of interactive spreadsheet builder. Thanks to the Opendata.cz initiative this data sets are made available as LOD modelled by the Daca Cube Vocabulary in the initiative's catalogue. The data is also hosted as a SPARQL endpoint[15]. 8 of these data sets have a dimension of reference period. Each data set's dump file can be downloaded from the catalouge[16].

```
@prefix qb: <http://purl.org/linked-data/cube#> .
@prefix czso: <http://data.czso.cz/ontology/> .

<http://data.czso.cz/resource/observation/job-applicants-and-unemployment-rate/CZ0513/2009-12-31/T> a
     qb:Observation ;
    czso:refArea <http://ruian.linked.opendata.cz/resource/okresy/3505> ;
    czso:refPeriod <http://reference.data.gov.uk/id/gregorian-day/2009-12-31> ;
    czso:sex sdmx-code:sex-T ;
    czso:neumisteniUchazeciOZamestnani 9692.0 ;
    czso:dosazitelniNeumisteniUchazeciOZamestnani 9528.0 ;
    czso:podilNezamestnanych 7.92 ;
    czso:pocetVolnychMist 569.0 ;
    qb:dataSet <http://data.czso.cz/resource/dataset/job-applicants-and-unemployment-rate> .
```

Listing 7.5: Example of an observation from the CZSO data sets

Observations in CZSO data cubes are assigned multiple measures. Their URIs are not dereferenceable. Their dimension value URIs do not work as proxy entities. The dimension of reference area uses entities of an above-mentioned initiative's unofficial RTIAR transformation[17] with its own SPARQL endpoint[18]. The measured values relate to regions and district. They do not contain observations related to the whole state. The proxy entities of the reference area dimension values for the CSSA data sets previously linked to this code list.

For time intervals representation the CZSO data cubes also use the `data.gov.uk` Time Intervals OWL ontology. They only do so directly unlike the CSSA data cubes. The data cubes vary in their time span. The earliest recorded values are for the year 2005. The latest values are for the year 2013. There are two data sets that contain values for both the earliest and latest year mentioned meaning they cover a period of 9 years: **czso-deaths-by-selected-causes-of-death**[19] and **czso-job-applicants-and-unemployment-rate**[20]

[14] https://vdb.czso.cz/vdbvo2/faces/en/index.jsf?page=uziv-dotaz

[15] http://linked.opendata.cz/sparql

[16] The download link URLs are, however, broken and return HTTP status code 404. To get the dump file, word *dumps* has to be substituted with word *soubor* (czech word for *file*). For example, the dump file of the data set **czso-job-applicants** is available at https://linked.opendata.cz/soubor/czso-job-applicants.trig

[17] https://linked.opendata.cz/dataset/cz-ruian

[18] https://ruian.linked.opendata.cz/sparql

[19] https://linked.opendata.cz/dataset/czso-deaths-by-selected-causes-of-death

[20] https://linked.opendata.cz/dataset/czso-job-applicants-and-unemployment-rate

Just as with the CSSA data cubes, some of the CZSO data cubes contain the dimension of sex consisting of three distinct values: dimension of male pension, dimension of female pensions and its total. The values used are the SDMX representations of sexes themselves.

7.3 Wikidata

Wikidata data set also contains data about political representation of countries, their administrative areas and municipalities. For the purposes of this experiment, such data concerning the Czech Republic was extracted from the data set. In the Czech Republic, regions and municipalities[21] are being assigned a government that emerges from elections. In Wikidata data set, there exist records of who was or still is head of this local government, including the head of the state government. Records of these *head of government* roles are given a time period of validity of this role by stating the date of this role's start and optionally the end of this role when it is not a current area government head anymore. For the persons who hold or held the office, the affiliation to a political party is stated in the data also with the start and end date of this affiliation. The entities of the political parties are assigned their political alignment (left, center, far-right, etc.). In the sample of Wikidata data set's content below it is stated that since 2008 till 2016 the head of the government of the South Moravian Region was Michal Hašek who since 1998 is a member of the Czech Social Democratic Party, which has the centre-left political alignment.

```
@prefix wd: <http://www.wikidata.org/entity/> .
@prefix p: <http://www.wikidata.org/prop/> .

wd:Q192697 rdfs:label "South Moravian Region"@en ;
    p:P6 [ p:P6 wd:Q6835752 ; p:P580 "2008-11-21T00:00:00Z" ; p:P582 "2016-11-16T00:00:00Z" ] .

wd:Q6835752 rdfs:label "Michal Hašek"@en ;
    p:P102 [ p:P102 wd:Q341148 ; p:P580 "1998-01-01T00:00:00Z" ] .

wd:Q341148 rdfs:label "Czech Social Democratic Party"@en ;
    p:P1387 [ p:P1387 wd:Q737014 ] .

wd:Q737014 rdfs:label "centre-left"@en .
```

Listing 7.6: Deluge of Wikidata triples

When adequately preprocced, this data can be utilized to find relations of statistical data described in the data cubes of CSSA and CZSO and the political cycle in the country. For example a rule can be found that states, that if in any year, the head of the Czech Republic's government was a member of a left-leaning policital party, the pension expenses for one-off allowance to pensions were above average. A query that extracts this data from the Wikidata's SPARQL endpoint is listed in A.1. This data, however, cannot be used for mining such rules yet. Measures in the data cubes are recorded on year to year bases. For the governmental roles and political affiliations it is only know the start date and end date.

[21] Not districts though.

It is necessary to transform these triples into set of triples stating that a governmental role or the policital affiliation *applies* for a certain year. The edge years are a bit tricky because the role or the membership was not valid for the whole year it started or ended. To facilitate the query and to generate more triples I decided to generate the triples for the edge years as well. The SPARQL query that constructs the *appliesToRefPeriod* triples from the extracted data is listed in A.2. The triples stating the start dates and end dates are no longer needed and do not have to be loaded into the RDFRules mining task.

The triples with predicates of `p:P582` and `p:P580` are no longer needed and are filtered out during the preproccesing. That will leave the data with 24 410 distinct triples.

7.4 YAGO Triples

Triples concerning the entitities of reference areas in the examined data cubes can also be found in the YAGO[22] dataset, which extracts facts about real world entities from various sources (Wikipedia, GeoNames, WordNet, etc.). For a change unlike with the Wikidata dataset, for the experiments involving the this data the needed triples are not extracted according to a rigidly structured CONSTRUCT queries but with loose DESCRIBE queries returning all triples concerning the sought entities. The used queries[23] extract the triples containing the reference areas themselves, the entities appearing in the triples together with the reference area entities (*hop 1*) and the entities appearing in the triples of those entities (*hop 2*). YAGO's public SPARQL endpoint's web interface[24] has trouble handling the volume of the triples returned in the latter mentioned queries so one is better off quering the endpoint directly through a HTTP client.

Both districts and regions of the Czech Republic have their class[25][26] of which they are subclass in the YAGO dataset so the needed triples are easy to query. Total of 2 992 361 distinct triples was extracted, containing data about persons connected to the areas, geographical information about the areas (for example which region contains which district), etc.

```
@prefix schema: <http://schema.org/> .
@prefix yago: <http://yago-knowledge.org/resource/> .

yago:Prague a yago:Capital_city .
yago:Josef_Jiří_Stankovský_Q12026167 schema:deathPlace yago:Prague .
yago:What_the_Old_Man_Does_is_Always_Right_Q13564487 schema:translator yago:Josef_Jiří_Stankovsk
    ý_Q12026167
```

Listing 7.7: Example of the Extracted YAGO Triples

[22] https://yago-knowledge.org/
[23] https://github.com/nvkp/diploma-thesis-code/tree/master/data/yago
[24] https://yago-knowledge.org/sparql
[25] http://yago-knowledge.org/resource/Districts_of_the_Czech_Republic
[26] http://yago-knowledge.org/resource/Regions_of_the_Czech_Republic

Around 62% of the extracted triples are not relevant for the performed tasks. Their objects are literals (`rdfs:label`, `rdfs:comment` and `schema:alternateName`) or image URLs (`schema:image`). These triples are filtered out before the index data structure is constructed.

7.5 Filtering the Observations

The values for the city of Prague are duplicated. The city is assigned an entity both as a region and as a district. The administrative area of the Czech Republic's capital is given a special status by the Act No. 131/2000 Coll., on the Capital City of Prague and does not in fact fall into neighter of those categories. Nonetheless, Prague is assigned an identifier both as a district (3 100) and as a region (19) in the RTIAR registry. When it comes to total population of the area (1 324 227 as of 2020), it is comparable to other czech regions. The least populated is the Karlovy Vary Region with around 300 000 inhabitants and the most populated: the Central Bohemian Region has around 1 300 000 inhabitants. Its surface area is on the other hand comparable with the districts. As the statistics about pensioners are certainly more correlated with the population rather than the surface area, the observation allocated to the dimension value of Prage as being district would be filtered out to maintain the commensurability along values measured for districts (see 7.6).

I also chose to filter out all observations regarding year 2008. For 2008 the pension kinds are structured according to a different scheme than any other year and it is hard to assume compatibility for the URIs that only differ in the year's suffix. 2008 scheme contains penkind kinds that 2010 does not end vice versa. It could be possible to just cut the cube so that the year's 2008 become one cube and the other years the other one, but a cube concerning only one reference period has not got much value. Both filters can be performed in a single SPARQL query. In the *Pensions* data set, discarting the Prague as a District entity removes 3 609 observations. The year 2008 contained 28 458 observations. 336 330 observations are contained in the query's result making up 91,4% of the unfiltered data set.

7.6 Slicing the Cubes

It is aimed to mine rules, in which the head atom's predicate is one of the cube's measures. In order to ensure, that such rules can achieve a reasonable support, the numerical values at the position of object in the measure triples have to be discretized and replaced by intervals. Irrespective to a chosen discretization approach, it is inadmissible to discretize values belonging to different disproportionate contexts (see 6.1.4). For example, we cannot create intervals for the number of pensioners from values measured for both regions and districts together. A district is a lower administrative unit. It belongs to a lower level in the concept hierarchy and it is assumed that its numbers of pensioners are of a different order of magnitude than those for regions or for the whole state. Same applies for values of dimensions sex and pension kind

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX qb: <http://purl.org/linked-data/cube#>
PREFIX cssa-d: <https://data.cssz.cz/resource/dataset/>
PREFIX cssa-od: <https://data.cssz.cz/ontology/dimension/>
PREFIX cssa-rd: <https://data.cssz.cz/resource/ruian/okresy/>
PREFIX cssa-op: <https://data.cssz.cz/ontology/pension-kinds/>

CONSTRUCT {
    ?observation ?p ?o
}
WHERE {
    GRAPH cssa-d:duchodci-v-cr-krajich-okresech {
        ?observation qb:dataSet cssa-d:duchodci-v-cr-krajich-okresech ;
            cssa-od:druh-duchodu ?druh ;
            ?p ?o .
        NOT EXISTS {
            ?observation cssa-od:refArea cssa-rd:3100 .
        }
     }
     GRAPH cssa-d:pomocne-ciselniky {
        ?druh skos:inScheme cssa-op:PensionKindScheme_2010 .
     }
}
```

Listing 7.8: SPARQL query to filter the *Pensions* data set

of the described data set. The values of the reference period represent even time intervals so the commensurability is assumed.

The chosen way to solve this is to slice the preprocessed cubes having disproportionate dimension values into sets of smaller subcubes, in which the dimension values belong to the same level of a concept hierarchy. Measured values were then discretized into intervals in each subcube separately. Also when the commensurability could not be expected among dimension values on the same level of their hierarchy, A cut was made for each dimension value. For example the *Pensions* data set had to be divided into 333 subcubes. It has 37 pension kinds. Dimension of sex has three distinct values: each sex and total. The dimension of reference area is considered to have 3 hierarchy levels: State's total, regions and Prague districts combined with the regional districts since they are comparable in number of inhabitants.

$$37 \times 3 \times 3 = 333 \ \textit{subcubes}$$

The dataset `cssa-d:vydaje-na-duchody-v-cr`[27] capturing costs on pensions in the Czech Republic by year and kind of pension contains 10 distinct values of the dimension of pension kind (not considering the scheme used only for year 2008). The cube had to be divided into 10 subcubes for each value of the pension kind dimension.

The construction of a subcube from a *master* cube can be performed by a SPARQL CONSTRUCT query. An example of such query is shown in the listing 7.9. This query filters the

[27] https://data.cssz.cz/resource/dataset/vydaje-na-duchody-v-cr

triples of the CZSO data cube **czso-job-applicants-and-unemployment-rate** to create a smaller cube of statistics about job applicants and unemployment rate for districts by sex. Notice how the reference area values corresponding to districts are distinguished. After the reference area values are linked (see 7.7) to their CSSA counterparts, the ontology provided with the CSSA data sets can be reused.

```
PREFIX qb: <http://purl.org/linked-data/cube#>
PREFIX sdmx-c: <http://purl.org/linked-data/sdmx/2009/code#>
PREFIX czso: <http://data.czso.cz/ontology/>
PREFIX czso-rd: <http://data.czso.cz/resource/dataset/>
PREFIX cssa-rd: <https://data.cssz.cz/resource/dataset/>
PREFIX cssa-or: <https://data.cssz.cz/ontology/ruian/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

CONSTRUCT { ?observation ?p ?o }
WHERE {
    GRAPH czso-rd:job-applicants-and-unemployment-rate {
        ?observation qb:dataSet czso-rd:job-applicants-and-unemployment-rate ;
            ?p ?o ;
            czso:refArea ?refAreaCZSO .
        NOT EXISTS {
            ?observation czso:sex sdmx-c:sex-T .
        }
     }
    ?refAreaCSSZ owl:sameAs   ?refAreaCZSO.
    GRAPH cssa-rd:pomocne-ciselniky { ?refAreaCSSZ a cssa-or:Okres }
}
```

Listing 7.9: SPARQL query to create a subcube

For every subcube a similar query has to be created and performed over the master cube. For a cube that has to be divided into a small number of subcubes it is plausible to write (and save it for the documentation and repeatability purposes) and perform these queries manually. But there are cubes for which this would involve an hours long work. At the same time, it is an trivial activity that can easily be automated. For this preproccesing step for the *Pensions* dataset, a Scala script[28] was written that creates 333 distinct SPARQL queries that construct 333 subcubes, saves each query to a text file and also creates a shell script that triggers all queries and saves a result of each query to a distinct file in the turtle format. This, however, still requires writing such script for each preprocced data cube and solve the problem of a time consuming resolution of the queries.

The URIs of the data set to which the observations pointed with the *qb* : *dataSet* had to be change to a *subcube specific* URI so that the observations in a rule can be anchored to a single subcube for a variable.

[28] https://github.com/nvkp/diploma-thesis-code/blob/master/data/pensions/script.scala

7.7 Linking

In order to find rules describing relations across multiple sources (meaning data cubes of CZSO, data cubes of CSSA, Wikidata and YAGO triples) the entities either have to be assigned the same URIs or to be connected by the `owl:sameAs` statements. The shared dimensions of the CSSA and CZSO data cubes are the reference area, reference period and sex. The dimension values URIs used for these dimensions differ not only institution from institution but also data cube from data cube from the same institution (In the CSSA data cubes, reference period is represented by an entity of a year and of the last day of the year as well). Linking of equivalent dimension values of the three dimensions is done by creating `owl:sameAs` statements.

7.7.1 Sex Dimension Values

It was already mentioned that the CSSA data cubes use proxy entities linking to the SDMX representations of sexes, whereas the CZSO data cubes use these representations directly. So the linking statements are already provided with the CSSA code list file. To extract these very triples, the query listed below can be used. These triples can be than loaded into a mining task involving mining from data cubes contain the dimension of sex instead of loading the whole code list file.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>

CONSTRUCT { ?cssaSex owl:sameAs ?sdmxSex }
WHERE {
    GRAPH <https://data.cssz.cz/resource/dataset/pomocne-ciselniky> {
        ?cssaSex a <https://data.cssz.cz/ontology/sdmx/code/Sex> ; owl:sameAs ?sdmxSex
    }
}
```

Listing 7.10: Linking the sex dimension values

7.7.2 Reference Periods

In CSSA data cubes, values from two concept schemes are used for representing the year intervals: a years scheme and a days scheme. The entities in the schemes are proxy entitities linking to the `data.gov.uk` Time Intervals ontology. CZSO data cubes use the ontology's day scheme concepts directly. That means that every year is represented by three distinct URIs in the data cubes so two `owl:sameAs` statements are required for each year. A query was written that generates these statements for every year entity in the CSSA code list:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

CONSTRUCT {
    ?cssaYear owl:sameAs  ?cssaDay .
    ?cssaYear owl:sameAs ?dataGovDay .
    ?cssaDay owl:sameAs ?dataGovDay .
}
WHERE {
    GRAPH <https://data.cssz.cz/resource/dataset/pomocne-ciselniky> {
        ?cssaYear skos:inScheme <https://data.cssz.cz/ontology/years/YearsScheme>
        BIND (REPLACE(str(?cssaYear),".*(\\d{4})","$1") as ?cssaYearValue)
        ?cssaDay skos:inScheme <https://data.cssz.cz/ontology/days/DaysScheme>
        FILTER (REGEX(str(?cssaDay), ".*day.*12-31"))
        BIND (REPLACE(str(?cssaDay),".*(\\d{4})-12-31","$1") as ?cssaDayValue)
        ?cssaDay owl:sameAs ?dataGovDay
        FILTER (?cssaYearValue = ?cssaDayValue)
    }
}
```

Listing 7.11: Linking the reference periods

7.7.3 Reference Areas

The proxy entities of the reference area in CSSA data set used to link to the same entities that are used by the CZSO data sets. This linking is no longer present in the CSSA's code list but can be retrieved from the Opendata.cz's SPARQL endpoint. The query listed below returns 114 linking triples for all districts (including the Prague district entity), all regions (including Prague), Prague districts and the entity of the whole state.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>

CONSTRUCT {
    ?cssaArea owl:sameAs ?odArea
}
WHERE {
    ?cssaArea a ?class ; owl:sameAs ?odArea .
    FILTER (?class IN (
        <https://data.cssz.cz/ontology/ruian/Okres>,
        <https://data.cssz.cz/ontology/ruian/Vusc>,
        <https://data.cssz.cz/ontology/ruian/SpravniObvod>,
        <https://data.cssz.cz/ontology/ruian/Stat>
        )
    )
}
```

Listing 7.12: Linking the reference areas of CSSA and CZSO

The linking with the Wikidata's entities had to be performed manually. But the YAGO data set already contains the *owl* : *sameAs* triples linking to Wikidata, so it was easy to extract those triples by a second SPARQL query and *close the linking circle*.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX yago: <http://yago-knowledge.org/resource/>
PREFIX schema: <http://schema.org/>

construct {
?region owl:sameAs ?sameRegion .
?district owl:sameAs ?sameDistrict .
}
where {
    ?region a yago:Regions_of_the_Czech_Republic .
    OPTIONAL {
        ?region schema:containsPlace ?district.
        ?district a yago:Districts_of_the_Czech_Republic .
        ?district owl:sameAs ?sameDistrict .
        FILTER REGEX(str(?sameDistrict), ".*http://www.wikidata.org/entity/.*")
    }

    ?region owl:sameAs ?sameRegion .
    FILTER REGEX(str(?sameRegion), ".*http://www.wikidata.org/entity/.*")
}
```

Listing 7.13: Linking the reference areas of YAGO and Wikidata

7.8 Discretization

Only the equifrequent discretization was chosen for the purposes of this experiment. Multiple discretizations (different interval counts) were performed with different parameters. If the sizes of the subcubes varied a lot, different sets of discretizations (interval counts) were set for them, because very coarse intervals worked fine for smaller subcubes, but with the same intervals the number of rules generated from the bigger subcubes was too high to be worked with even when the timeout threshold was set.

7.9 Mining Tasks

The source codes of the performed experiment are available in repository on github[29] in the directory *notebooks* in a form of jupyter[30] notebooks. A Scala kernel for jupyter (eg. *almond.sh*[31]) has to be install in order to execute the code directly from the notebooks (see *README.md*). Some of the notebooks are dedicated to preproccesing of a particular data set and other contain the performed mining tasks. In the preproccesing notebooks the preprocced data set is exported to a text file in the *Turtle* format and also saved into a cache file from which the data set is loaded into memory before a mining task. Github strictly enforces 100 MB size limit for each file so the cache files and *Turtle* files are not pushed into the repository. In order to run the mining tasks locally one has to create the cache files by running the code in the preproccesing notebooks.

[29] https://github.com/nvkp/diploma-thesis-code
[30] https://jupyter.org/
[31] http://almond.sh/

7.9.1 Relation between the pension expenses and the policital alignment of the state's government

A relation can be assumed between the expenditure on pensions in a state social security system and the political ideology of the state. A left leaning governments usually tend to spend more on social welfare than the right-wing governments. In this section it is shown how this relation can be described by association rules in the spirit of 4.1 mined from the data published by CSSA and Wikidata. Data from various countries would be more appropriate for mining such relation. The available data are, however, sufficient for a simple demonstration.

The data cube used[32] is published by CSSA and contains records of total annual expenses on particular pensions. From now on it will be denoted as *expenses*. The only measure of the data cube is the expenditure amount in thousands of CZK. The dimensions of the data cube are the reference period with the granularity of whole years (from 2008 till 2019) and the dimension of the pension kind containing individual pension kind and also aggregated dimension values. For the year 2008 and 2009 and old pension scheme is used so the observations considering these two years were discarded. For the dimension of reference period 10 distinct values remain and the newer pension kind scheme contains also 10 distinct values. That would suggest that 100 observations remain but there are only 94. For the one-off allowance to pensions there are only 4 observations. That probably means that the expenses for this pension kind in the missing years were zero. So the missing observations with the measured value of zero were created manually.

The original data cube was sliced into 10 subcubes (by SPARQL queries) each containing 10 observations. Each subcube is in a separate file in the folder `data/expenses`. In the notebook `expenses.ipynb`[33] each subcube's measures are discretized equifrequently (with 2 and 5 intervals) and equisizeably (with the relative support of 20, 30 and 50 percent) assigning each measurement 5 intervals.

The preproccesing of the Wikidata data set (notebook `wikidata.ipynb`[34]) rested in merging the triples returned by queries A.1 and A.2 and removing the triples with predicates `P580` and `P582` as they only served to generate derived triples in the query A.2.

In the notebook (`expenses-wikidata.ipynb`[35]) of the mining task the two preprocessed data sets were loaded and merged together with the reference period linking triples. From this data set containing 25 494 triples an instance of index was created. The rules that should be found are described in plain text as *If in any year the current prime minister belongs to a political party that has a certain political alignment then the annual expenses of a certain pension kind fit into certain interval.*

[32] https://data.cssz.cz/web/otevrena-data/-/vydaje-na-duchody-v-cr
[33] https://nvkp.github.io/diploma-thesis-code/notebooks/expenses
[34] https://nvkp.github.io/diploma-thesis-code/notebooks/wikidata
[35] https://nvkp.github.io/diploma-thesis-code/notebooks/expenses-wikidata

Let's write it as a *pseudo* rule pattern in the context of the available data's structure:

$$(?headRole\ appliesTo\ ?refPeriod)(\langle Czech\_Republic\rangle\ headOfGovernment\ ?headRole)$$

$$\wedge(?headRole\ person\ ?person) \wedge (?person\ partyRole\ ?partyRole)$$

$$\wedge(?party\ alignment\ AnyConstant) \wedge (?partyRole\ party\ ?party)$$

$$\wedge(?partyRole\ appliesTo\ ?refPeriod) \wedge (?observation\ refPeriod\ ?refPeriod)$$

$$\wedge(?observation\ dataSet\ AnyConstant) \Rightarrow (?observation\ expenses\ AnyConstant)$$

The variable ?*headRole* would be instantiated by entities representing an acting of a specific person in the office just as the variable ?*partyRole* would be instantiated by entities of memberships of a specific person in a specific political party. The rules yielded from this rule pattern would differ only by the entities at the object positions of the fifth and nineth atom and the head atom. The last body atom with the `qb:dataSet` will ensure that all ?*observation* bindings belong to the same sub cube and thus the commensurability of measures will be preserved. There is no need of an atom stating the pension kind, because all observations in a sub cube share the same one. That way the rule is shorter.

The used algorithm implementation does not allow custom names for variables in a pattern. The variable names have to be single characters ordered alphabeticaly beginning with ?*a* from the rule's head. The definition of the rule pattern object to match the sought relations in this mining task is shown in listing 7.14:

```
val alignmentExpenses: RulePattern = (
    AtomPattern(subject = 'f', predicate = appliesTo, `object` = 'b') &:
    AtomPattern(subject = czURI, predicate = wdProperty(6), `object` = 'f') &:
    AtomPattern(subject = 'f', predicate = wdProperty(6), `object` = 'e') &:
    AtomPattern(subject = 'e', predicate = wdProperty(102), `object` = 'c') &:
    AtomPattern(subject = 'd', predicate = wdProperty(1387)) &:
    AtomPattern(subject = 'c', predicate = wdProperty(102), `object` = 'd') &:
    AtomPattern(subject = 'c', predicate = appliesTo, `object` = 'b') &:
    AtomPattern(subject = 'a', predicate = cssaRefPeriod, `object` = 'b') &:
    AtomPattern(subject = 'a', predicate = qbdPredicate, `object` = AnyConstant)
    =>:
    AtomPattern(subject = 'a', predicate = expenses)
)
```

Listing 7.14: First pattern definition

This pattern was connected to the mining task definition itself. No minimal support an head coverage thresholds were set so that a maximal number of rules that can be filtered later are found. The maximal rule length set corresponds to the provided rule pattern. The mining task generated 96 rules[36]. The rules with a support of 1 were filtered out and the PCA confidence, the standard confidence and lift was computed on the remaining 58 rules[37]. The listing 7.17 shows two of them as they are written in the export file.

[36] https://nvkp.github.io/diploma-thesis-code/rulesets/alignmentExpensesTaskRuleset.txt
[37] https://nvkp.github.io/diploma-thesis-code/rulesets/alignmentExpensesTaskRulesetFiltered.txt

```
(?f prfx:appliesToRefPeriod ?b) ^ (wd:Q213 p:P6 ?f) ^ (?f p:P6 ?e) ^ (?e p:P102 ?c) ^
    (?d p:P1387 "centre-right") ^ (?c p:P102 ?d) ^ (?c prfx:appliesToRefPeriod ?b) ^
    (?a cssa-od:refPeriod ?b) ^ (?a qb:dataSet <expenses-old-age-total>) -> (?a cssa-
    om:vydaje-na-duchody-opravene-o-zalohy-v-tis-kc <<3.1797095155E8__3.8222294828E8)
    _ef3_3/3>)
| support: 3, headCoverage: 0.005, confidence: 1.0, pcaConfidence: 1.0, lift: 25.0,
    headConfidence: 0.04, headSize: 600, bodySize: 3, pcaBodySize: 3

(?f prfx:appliesToRefPeriod ?b) ^ (wd:Q213 p:P6 ?f) ^ (?f p:P6 ?e) ^ (?e p:P102 ?c) ^
    (?d p:P1387 "centrism") ^ (?c p:P102 ?d) ^ (?c prfx:appliesToRefPeriod ?b) ^ (?a
    cssa-od:refPeriod ?b) ^ (?a qb:dataSet <expenses-it>) -> (?a cssa-om:vydaje-na-
    duchody-opravene-o-zalohy-v-tis-kc <<2.488316469E7__2.553204263E7)_ef3_1/3>)
| support: 2, headCoverage: 0.0033333333333333335, confidence: 0.6666666666666666,
    pcaConfidence: 0.6666666666666666, lift: 22.22222222222222, headConfidence: 0.03,
    headSize: 600, bodySize: 3, pcaBodySize: 3
```

Listing 7.15: First pattern's rule sample

The first rule states that when the government has the center-right political alignment, the total expenses for all types of old age pensions are in the upper third. The second rule predicts the lower third value of expenses on the third degree invalidity pension, when the government's policital alignment is centristic.

Note that as the section 6.1.4 describes, the head size values are distorted, because the observations's measures are structure by the style A and the whole cube is mined at once. Each of the 100 observations appears in the head size as many times as there are measure intervals assigned to the head measure predicate. This also distorts the head coverage. Also the lift is over estimated because its denominator works with all 100 observations when the only observations that matter are that of the sub cube specified in the rule's body.

Quite suprisingly in none of the 96 rules the left alignment is mentioned even though Czech Republic's prime minister from the years 2014 to 2017 was a left leaning politician. Only alignments mentioned are *centre-right* and *centrism*. The only governing party assigned to those values in the covered period is *ANO 2011*[38]. It seems that the party itself is a sufficient *explanatory variable* for the measure in the data cube. So the pattern was simplified to connect the measured values only to the appropriate party.

29 rules matching the pattern in 7.16 exceed support of 1. Only party appearing in these rule is the above mentioned *ANO 2011* (`wd:Q10728124`). It is the latest governing party in the covered period so it could seem that, more than the relation between the pensions expenditure and governing party, the rules describe a relation between pensions expenditure and time because the measured values are not treated for inflation. There are, however, also rules that predict the lowest of the intervals.

[38] https://www.wikidata.org/wiki/Q10728124

```
val partyExpenses: RulePattern = (
    AtomPattern(subject = 'e', predicate = appliesTo, `object` = 'b') &:
    AtomPattern(subject = czURI, predicate = wdProperty(6), `object` = 'e') &:
    AtomPattern(subject = 'e', predicate = wdProperty(6), `object` = 'd') &:
    AtomPattern(subject = 'd', predicate = wdProperty(102), `object` = 'c') &:
    AtomPattern(subject = 'c', predicate = wdProperty(102), `object` = AnyConstant) &:
    AtomPattern(subject = 'c', predicate = appliesTo, `object` = 'b') &:
    AtomPattern(subject = 'a', predicate = cssaRefPeriod, `object` = 'b') &:
    AtomPattern(subject = 'a', predicate = qbdPredicate, `object` = AnyConstant)
    =>:
    AtomPattern(subject = 'a', predicate = expenses)
)
```

Listing 7.16: Second pattern definition

```
(?e prfx:appliesToRefPeriod ?b) ^ (wd:Q213 p:P6 ?e) ^ (?e p:P6 ?d) ^ (?d p:P102 ?c) ^
    (?c p:P102 wd:Q10728124) ^ (?c prfx:appliesToRefPeriod ?b) ^ (?a cssa-od:refPeriod
     ?b) ^ (?a qb:dataSet <expenses-old-age-total>) -> (?a cssa-om:vydaje-na-duchody-
    opravene-o-zalohy-v-tis-kc <<3.1797095155E8__3.8222294828E8)_ef3_3/3>)
| support: 3, headCoverage: 0.005, confidence: 1.0, pcaConfidence: 1.0, lift: 25.0,
    headConfidence: 0.04, headSize: 600, bodySize: 3, pcaBodySize: 3

(?e prfx:appliesToRefPeriod ?b) ^ (wd:Q213 p:P6 ?e) ^ (?e p:P6 ?d) ^ (?d p:P102 ?c) ^
    (?c p:P102 wd:Q10728124) ^ (?c prfx:appliesToRefPeriod ?b) ^ (?a cssa-od:refPeriod
     ?b) ^ (?a qb:dataSet <expenses-it>) -> (?a cssa-om:vydaje-na-duchody-opravene-o-
    zalohy-v-tis-kc <<2.488316469E7__2.553204263E7)_ef3_1/3>)
| support: 2, headCoverage: 0.0033333333333333335, confidence: 0.6666666666666666,
    pcaConfidence: 0.6666666666666666, lift: 22.22222222222222, headConfidence: 0.03,
    headSize: 600, bodySize: 3, pcaBodySize: 3
```

Listing 7.17: Second pattern's rule sample

7.9.2 Appending the YAGO Data Set to the CZSO Data Cubes

In this task the mining tasks were performed over the CZSO's data cube *Job Applicants and Unemployment Rate*, from now on denoted as *JAUR*, and the triples extracted from YAGO data set. The data cube contains dimensions of reference area (89 distinct dimension values constisting of 13 regions without Prague and 76 districts without Prague, meaning there are no data concerning Prague), reference period (years 2005 to 2013) and sex (male, female and total). Each observation in the cube contains multiple measures. All observations are assigned the measures of number of available job applicants (job applicants who have no objective obstacle to taking up a job, e.g. enrollment in retraining courses or serving a sentence) and unemployment rate. Observations concerning both sexes also contain the measures of all job applicants and number of vacancies.

The data cube had to be sliced along the dimensions of reference area and sex. The dimension of reference area was divided into regions and districts. It was considered to divide the dimension of sex by each dimension value, but there is no significant difference between the male and female values[39], so the dimension was divided into *total* and *by-sex* part. That

[39] https://nvkp.github.io/diploma-thesis-code/data/jaur/SexDimension

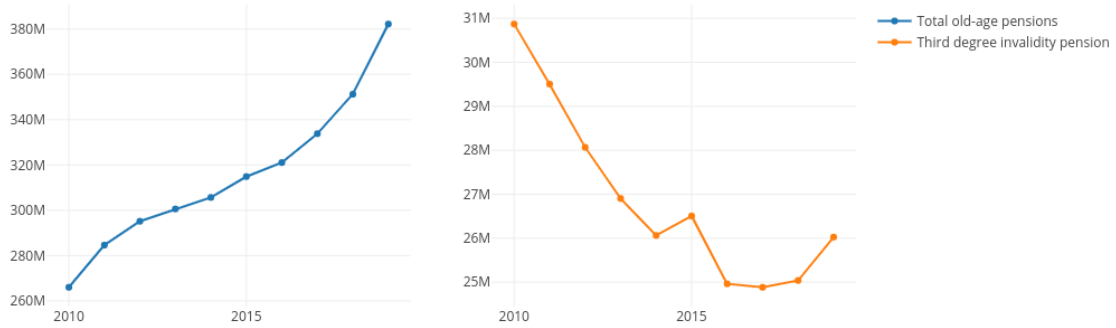

Figure 7.1: Expenditure of selected pension kinds over time

makes 4 slices [40] with 2 403 observations ($3 \times 9 \times 89$) and 6 408 measured values in total, here denoted as *regions-total*, *regions-by-sex*, *districts-total* and *districts-by-sex*

All the measured values were discretized[41] equifrequently. The measures of the subcubes *regions-total* and *regions-by-sex* were discretized twice with interval counts of 5 and 10. The measures of the subcubes *districts-total* and *districts-by-sex* were discretized with interval counts of 10, 20 and 30.

The notebook `jaur-yago.ipynb`[42] contains the mining itself. The YAGO triples and data cube's triples were merged together with the appropriate reference area linking triples[43]. A separate index (containing all the extracted YAGO triples, the reference area linking triples and only those triples from the *JAUR* cube that belong to the subcube's observations) was created for each of the four subcubes. That way the lift measure calculated for the rules mined from the indices is not distorted because its denominator is not increased by inappropriate observations.

Just as in the previous example, a pattern had to be provided to the mining tasks, that generates rules assigning a particular interval of a measure for observations satisfying the rule's body. In this pattern a variable at the position of object with the predicate of `refArea` appears in an atom that is enforced to bind to triples from the YAGO data set. The atoms of the YAGO are enforced only to be present. They are not prescribed any rigid structure in the rules. For a mining task working with such pattern the threshold of maximal rule's length can be declared that corresponds to the number of hops of the extracted YAGO triples (determining the maximal length of the atom `chain` from YAGO) and the number of other atoms in the body and the head atom.

[40] https://nvkp.github.io/diploma-thesis-code/data/jaur/slice-queries/
[41] https://nvkp.github.io/diploma-thesis-code/notebooks/jaur
[42] https://nvkp.github.io/diploma-thesis-code/notebooks/jaur-yago
[43] https://nvkp.github.io/diploma-thesis-code/data/linking/yagoCZSOLinking.ttl

For fininishing the mining task in a reasonable time also a minimal support thresholds had to be declared. The support of the sought rules corresponds to the number of observations for which the rule is valid. The rules have to contain an atom achoring the observations' variable to a constant of a slice by the `qb:dataSet` predicate to ensure that each rule relates to one and only slice. The slices of the *JAUR* data cube contain a different number of observations, so if a too high minimal support threshold is declared, the potentionally accurate rules from the a smaller slices would be discarded. In 7.9.1 all the slices contain the same number of observations and the whole cube so small that no minimal support threshold had to be used. In this mining task the problem was avoided by creating 4 distinct patterns that anchor the observations to each slice. These patterns are then appended to a separate mining tasks for each subcube with different minimal support thresholds.

```
val districtBySexPattern = (
    AtomPattern(subject='b',graph=uri("yago"))&:
    AtomPattern(subject='a',predicate=refArea,`object`='b',graph=uri("czso"))&:
    AtomPattern(subject='a',predicate=qbDataSet,`object`=districtBySexSlice,graph=uri(
        "czso"))
    =>:
    AtomPattern(subject='a',predicate=oneOfMeasures, graph = uri("czso"))
)
```

Listing 7.18: Pattern definition for the slice *districts-by-sex*

The smallest slice is `jaur-regions-total` with only 117 observations (9 years of 13 regions). Let's assume there is a rule that can be inferred from this data that predicts a certain measure value to be in a certain interval given a certain characteristic of the reference area of the observation. Not to describe only one specific region, there have to be at least two regions satisfying the rule. If the rule had the confidence of 1 its support would be 18 (9 years $\times$ 2 regions). A rule with confidence of 0,5 would have half the support. Less confident rules with support of 9 or higher support would *cover* more regions. So the minimal support for each mining task corresponding to single slice was given as the number of observations in the slice divided by number of observations for a distinct reference area. For the district slices such designated threshold would however not restrict the search space to finish the procedure in a reasonable time so for these two mining tasks with largest number of observations the number was multiplied by 3 (We demand *half confident* rules to concern at least 6 districts).

```
val districtBySexTask = Amie()
    .addThreshold(Threshold.MinSupport(minSupport(districtBySexSlice)*3))
    .addThreshold(Threshold.MaxRuleLength(6))
    .addConstraint(constantsOnlyAtObject)
    .addPattern(districtBySexPattern)
```

Listing 7.19: Mining task definition for the slice *districts-by-sex*

RDFRules API provides a constraint that can be appended to the mining task that eliminates atoms with a constant at the position of subject to be considered during the rule refinement. This shrinks the task's search space and contributes to shorter runtime when such atoms are not relevant for the particular pattern. In 7.9.1 this constraint could not be used since the

second atom pattern's subject in the rule pattern was the constant of the Czech Republic's URI. It makes sense to add the constraint now.

Folows the description of the processing of each mining task's retrieved rule set. The used patterns do not prescribe a structure for each atom in a rule. The length of the rules can vary from 4 to 6 atoms. The patterns do not prevent the refining operators from appending an atom to the rule's body, whose subject variable represents a different observation than the one in the rule's head. That practically introduces a new *unclosed* cube to the rule and invalidates it. Atoms with a new observation variable in those rules are connected through the variable of reference area. They all contain atom with the new variable at the position of subject, reference area dimension URI at the position of predicate and the variable of reference area at the position of object. All those rules were filtered out. Only those rules in the rule set, that have exactly one atom with reference area dimension URI at the position of predicate were kept for further processing.

```
(?c czso:pocetVolnychMist <[ 1222.0 ; 1608.5 )_ef20_19/20>) ^ (?c czso:refArea ?b) ^
    (?b <http://schema.org/containedInPlace> yago:Moravian-Silesian_Region) ^ (?a czso
    :refArea ?b) ^ (?a qb:dataSet <jaur-districts-total>) -> (?a czso:
    neumisteniUchazeciOZamestnani <[ 9877.0 ; 26549.0 ]_ef10_10/10>)

| support: 30, headCoverage: 0.014619883040935672, confidence: 0.7142857142857143,
    pcaConfidence: 0.7142857142857143, lift: 6.979591836734694, headConfidence:
    0.1023391812865497, headSize: 2052, bodySize: 42, pcaBodySize: 42
```

Listing 7.20: Example of a rule with an unclosed body cube

There is apparently a correlation between the cube's measures (available applicants is a subset of all job applicants, lower unemployment rate corresponds to lower number of vacancies). The algorithm tends to create evident rules in the sense of *if there is a lower unemployment rate in the area, there is a lower number of applicants registered at the Labor Office*. Not to bother with this kind of rules, the rules with a measure URI at the position of predicate in any atom in the rule's body were filtered out as well.

```
(?a czso:dosazitelniNeumisteniUchazeciOZamestnani <[ 9639.0 ; 25767.0 ]_ef10_10/10>) ^
     (?b <http://schema.org/containedInPlace> yago:Moravian-Silesian_Region) ^ (?a
    czso:refArea ?b) ^ (?a qb:dataSet <jaur-districts-total>) -> (?a czso:
    neumisteniUchazeciOZamestnani <[ 9877.0 ; 26549.0 ]_ef10_10/10>)

| support: 30, headCoverage: 0.014619883040935672, confidence: 1.0, pcaConfidence:
    1.0, lift: 9.771428571428572, headConfidence: 0.1023391812865497, headSize: 2052,
    bodySize: 30, pcaBodySize: 30
```

Listing 7.21: Example of an rule describing the correlation of measures

The lift and confidence were computed for the remaining rules in the rule sets (each for every subcube). All rules with the lift value lower than 1 and the confidence value lower than 0,5 were filtered out. On the remaining rules in the rule sets the DBScan clustering algorithm was used with parameters of minimum size of a cluster of 3, minimum similarity of rules in the same cluster of 85%. The similarity was computed only based on the content of the

rules, not their interest measures. 673 rules from the *regions-total* ruleset were arranged into 75 clusters, 46 clusters emerged in the 457 rules of the *regions-by-sex* rule set, 169 rules of *districts-total* rule set were divided into 26 clusters, ant 320 rules of the *districts-by-sex* rule set into 28 of them. From those four rule sets new rule sets were derived, which contain one rule for each cluster. All rule sets, *raw*, filtered, clustered and pruned by clusters are published on a signpost page[44] in the repository's Github Pages.

7.9.3 Relation Between Measures from Different Data Cubes

This section describes the demonstration of how the AMIE algorithm can be used to mine relations of measures from multiple data cubes described in 6.3. The mining was performed on a cube published by CZSO containing average and median salaries for the regions in Czech Republic, here denoted as *Salaries* and the CSSA's *Pensions* cube mentioned in 7.1. The *Salaries* cube is structured into 3 dimensions. The dimension of sex consists of 3 distinct values of male, female and both sexes. The dimension of reference area has 14 distinct values for 13 regions and Prague. The dimension of reference period contains 3 distinct values, meaning the data covers only three years, from 2010 to 2012. Each observation in the cube is assigned two measures of the average and the median salary. The cube had to be cut to 3 subcubes[45] for each value of the dimension of sex, because the measures assigned to each value's observations were assumed incommensurable[46]. Each of the subcubes[47] contains 42 observations. Each measure was discretized[48] four times with the interval counts of 2, 3, 5 and 7.

The *Pensions* cube also contains statistics for individual districts and the whole state but those observations could be omitted, because they would not appear in the rules, since the *Salaries* cube does not share those reference area dimension values. The same goes for its dimension of reference area. Only the years intersecting the *Salaries* cube are useful. The filtered cube was cut[49] along the dimensions of pension kind with 37 distinct values and sex with 3 distinct values. For each combination of pension kind and sex dimension value a subcube was created so the total number of used subcubes[50] is 111. Those subcubes contain the same number of observations as the *Salaries* subcubes. The *Pensions* cube contains three measures: the average amount of pension, the average age, and the number of persons but each observation is assigned only one measure, so there are exactly three observations for each combination of pension kind[51], reference area, reference period and sex. That has an impact on the lift measure and standard confidence (not on the PCA confidence thoughl), when the rule predicts an interval for the observation in this cube. The lift is overestimated

[44] https://nvkp.github.io/diploma-thesis-code/rulesets/jaur-yago/
[45] https://nvkp.github.io/diploma-thesis-code/data/salaries/slice-queries
[46] https://nvkp.github.io/diploma-thesis-code/data/salaries/SexDimension
[47] https://nvkp.github.io/diploma-thesis-code/data/salaries
[48] https://nvkp.github.io/diploma-thesis-code/notebooks/salaries
[49] https://nvkp.github.io/diploma-thesis-code/data/pensions/slice-queries
[50] https://nvkp.github.io/diploma-thesis-code/data/pensions
[51] This is valid for the filtered observations used for this task. The observations for the years before 2010 use a different pensions scheme.

and the standard confidence is underestimated. Each measure was discretized twice with interval counts of 2 and 3.

The discretization of the *Pensions* cube and the mining itself is contained in notebook `pensions-salaries.ipynb`[52]. Three different instances of index were created. Each index was constructed from the triples of one of the *Salaries* subcubes, 37 *Pensions* subcubes with the same sex dimension value as the *Salaries* subcube and the linking *owl* : *sameAs* triples for reference area, sex and reference period. Two rule pattern were defined. One for the rules that predict an interval for an observation from one of the *Pensions* subcubes based on the values in the *Salaries* subcube and the seconds defines the shape of rules that predict an interval for an observation in the *Salaries* subcube based on the values in one of the *Pensions subcubes*.

```
val salariesPensionsPattern: RulePattern = (
    AtomPattern(subject = 'c', predicate = oneOfSalariesMeasures, `object` =
        AnyConstant) &:
    AtomPattern(subject = 'c', predicate = czsoRefPeriod, `object` = 'd') &:
    AtomPattern(subject = 'a', predicate = cssaRefPeriod, `object` = 'd') &:
    AtomPattern(subject = 'c', predicate = qbDataSet, `object` = AnyConstant) &:
    AtomPattern(subject = 'c', predicate = czsoRefArea, `object` = 'b') &:
    AtomPattern(subject = 'a', predicate = cssaRefArea, `object` = 'b') &:
    AtomPattern(subject = 'a', predicate = cssaPensionKind, `object` = AnyConstant) &:
    AtomPattern(subject = 'a', predicate = qbDataSet, `object` = AnyConstant)
    =>:
    AtomPattern(subject = 'a', predicate = oneOfPensionsMeasures)
)

val pensionsSalariesPattern: RulePattern = (
    AtomPattern(subject = 'c', predicate = oneOfPensionsMeasures, `object` =
        AnyConstant) &:
    AtomPattern(subject = 'c', predicate = cssaPensionKind, `object` = AnyConstant) &:
    AtomPattern(subject = 'a', predicate = czsoRefPeriod, `object` = 'd') &:
    AtomPattern(subject = 'c', predicate = cssaRefPeriod, `object` = 'd') &:
    AtomPattern(subject = 'c', predicate = qbDataSet, `object` = AnyConstant) &:
    AtomPattern(subject = 'c', predicate = cssaRefArea, `object` = 'b') &:
    AtomPattern(subject = 'a', predicate = czsoRefArea, `object` = 'b') &:
    AtomPattern(subject = 'a', predicate = qbDataSet, `object` = AnyConstant)
    =>:
    AtomPattern(subject = 'a', predicate = oneOfSalariesMeasures)
)
```

Listing 7.22: Pattern definitions for relation between cubes

Application of these rule patterns with the maximum rule length results only in rules where both cubes' dimensions are closed. For each combination of rule pattern and index the algorithm was invoked with no minimal support threshold and maximum rule length corresponding to the length of the rule patterns. Each of the 6 rule sets was filtered to so that only the perfect rules remained. Those rules were sorted descendantly by support. A signpost[53] page in the repository provides link to those exported rule sets. The rules containing broader intervals have higher support than the rules with narrower intervals. The listing 7.23 shows

[52] https://nvkp.github.io/diploma-thesis-code/notebooks/pensions-salaries
[53] https://nvkp.github.io/diploma-thesis-code/rulesets/pensions-salaries/

one of the rules with the highest support and one one of the rules with the lowest support.

```
(?c czso:medianMzdy <[ 19229.0 ; 21214.0 )_ef2_1/2>) ^ (?c czso:refPeriod ?d) ^ (?a
    cssz-dimension:refPeriod ?d) ^ (?c qb:dataSet <salaries-total>) ^ (?c czso:refArea
     ?b) ^ (?a cssz-dimension:refArea ?b) ^ (?a cssz-dimension:druh-duchodu <https://
    data.cssz.cz/resource/pension-kind/PK_V_total_2010>) ^ (?a qb:dataSet <pensions-by
    -region-total-PK_V_total>) -> (?a <https://data.cssz.cz/ontology/measure/prumerna-
    vyse-duchodu-v-kc> <[ 10506.559275967385 ; 11475.861138653103 )_ef3_1/3>)
| support: 21, headCoverage: 0.002027027027027027, confidence: 0.3333333333333333,
    pcaConfidence: 1.0, lift: 37.53623188405797, headConfidence: 0.00888030888030888,
    headSize: 10360, bodySize: 63, pcaBodySize: 21

(?c czso:medianMzdy <[ 22602.0 ; 28392.0 ]_ef7_7/7>) ^ (?c czso:refPeriod ?d) ^ (?a
    cssz-dimension:refPeriod ?d) ^ (?c qb:dataSet <salaries-total>) ^ (?c czso:refArea
     ?b) ^ (?a cssz-dimension:refArea ?b) ^ (?a cssz-dimension:druh-duchodu <https://
    data.cssz.cz/resource/pension-kind/PK_S_2010>) ^ (?a qb:dataSet <pensions-by-
    region-total-PK_S>) -> (?a <https://data.cssz.cz/ontology/measure/prumerny-vek> <[
     68.24930566713338 ; 70.01816377599448 )_ef3_1/3>)
| support: 6, headCoverage: 5.791505791505791E-4, confidence: 0.3333333333333333,
    pcaConfidence: 1.0, lift: 38.37037037037037, headConfidence: 0.008687258687258687,
     headSize: 10360, bodySize: 18, pcaBodySize: 6
```

Listing 7.23: Two of the resulting perfect rules

The first rules states that the regions with a lower median salary (half of all regions) are also regions with a lower widow's and widower's pension. That makes sense considering that the widow's and widower's pensions are calculated based on the old age pension of the deceased partner and the old age pension of the partner was partly dependant on his or her salary. The second rule states that the two regions with the highest median salary are also regions with a lower average age of people receiving the old age pension. Note that the standard confidence is a third of the PCA confidence. That is because also the observation not assigned to the predicted measure are considered counterexamples and not only the third of them with the correct measure.

```
(?c czso:prumernaMzda <[ 26722.0 ; 27860.0 )_ef5_4/5>) ^ (?c czso:refPeriod ?d) ^ (?a
    cssz-dimension:refPeriod ?d) ^ (?c qb:dataSet <salaries-male>) ^ (?c czso:refArea
    ?b) ^ (?a cssz-dimension:refArea ?b) ^ (?a cssz-dimension:druh-duchodu <https://
    data.cssz.cz/resource/pension-kind/PK_ID_2010>) ^ (?a qb:dataSet <pensions-by-
    region-male-PK_ID>) -> (?a <https://data.cssz.cz/ontology/measure/prumerny-vek> <[
     48.615229885057474 ; 50.58006905532994 )_ef2_1/2>)
| support: 9, headCoverage: 8.687258687258687E-4, confidence: 0.3333333333333333,
    pcaConfidence: 1.0, lift: 24.666666666666664, headConfidence:
    0.013513513513513514, headSize: 10360, bodySize: 27, pcaBodySize: 9

(?c <https://data.cssz.cz/ontology/measure/prumerny-vek> <[ 48.615229885057474 ;
    50.58006905532994 )_ef2_1/2>) ^ (?c cssz-dimension:druh-duchodu <https://data.cssz
    .cz/resource/pension-kind/PK_ID_2010>) ^ (?a czso:refPeriod ?d) ^ (?c cssz-
    dimension:refPeriod ?d) ^ (?c qb:dataSet <pensions-by-region-male-PK_ID>) ^ (?c
    cssz-dimension:refArea ?b) ^ (?a czso:refArea ?b) ^ (?a qb:dataSet <salaries-male
    >) -> (?a czso:prumernaMzda <[ 26722.0 ; 27860.0 )_ef5_4/5>)
| support: 9, headCoverage: 0.05357142857142857, confidence: 0.28125, pcaConfidence:
    0.28125, lift: 1.3125, headConfidence: 0.21428571428571427, headSize: 168,
    bodySize: 32, pcaBodySize: 32
```

Listing 7.24: Example of two *mirroring* rules

The unfiltered rule sets generated from the same index but according to a different rule pattern have the same number of rules. That means that the rules are *mirroring* each other and stating that either *A is associated with B* or *A is associated with B*. An example of those mirroring rules is shown in the listing 7.24.

The rules have the same support but a different confidence. The average salary in the range of 26 722 CZK to 27 860 CZK strongly implies a lower half of the average age of people receiving the second degree disability pension, but it does work the other way around. The second rule's confidence measures have the same value, because the *Salaries* assigns all both measures to a single observation.

8. Discussion of the Results

The mining task 7.9.1 generated meaningful and interpretable rules and it served as a confirmation, that the considerations of the needed pre-proccessing steps and the sought shape of the rules were correct and it can serve as a demonstration of the possibilities of performing this type of analysis. More diverse rules would be found if the examined data cube contained also the data about other branches of the government spending, not just the pension expenditure.

The problem of commensurability and different ranges of the shared dimensions across the cube caused, that even a quiet massive data cube with multiple hierarchies in the dimensions, such as the *Pensions* cube, had to be cut into tiny subcubes of a couple of observations, which prevents the rules to achieve any significant support in comparison with the association rules mined over transactional tabular data.

For someone who sees such rules for the first time and does not know under what conditions the rules emerged, it would be very difficult to interpret them, especially when the rules concern multiple data cubes and when not all atoms in the rule contain human-readable identifiers (as it is the case with the task 7.9.1, where the rules contain QIDs and properties in the form of numerical identifiers prefixed by `P`). Transforming those identifiers into a human-readable could be very time consuming depending and the analyst would deprive him or herself of a possibility to merge new sources of triples using those identifiers as well.

When the rules do not contain atoms from the *unrestrained* knowledge graphs but only use other measures in the same cube or other cubes, the advantage of this approach is not leveraged and e.g. the correlation or regression analysis would do a better job describing the dependence of the measures on each other.

When no rigid structure of the rules is enforced or when the measures are discretized in an excessive number of ways, an overwhelming number of rules is generated. That happened in the task 7.9.2 when the number of rules generated for each index was in the tens of thousands. Those rules could not be easily filtered only by confidence or lift, because the rules with top-ranked rules were very similar to each other[1]. RDFRules framework implements the feature to cluster the rules, so only a few or one rule from each cluster could be kept, but clustering tens of thousands of rules takes hours which makes it impossible to work interactively with the framework.

So I got into a *vicious circle* where I wanted as small and diverse set of rules as possible and I could not make it diverse because it was not small enough and I could not make it too small either because it would not be diverse enough. The top-k approach was not the solution, because RDFRules implement top-k pruning only by the head coverage, which advantages rules with broader intervals. The data coverage pruning feature of the RDFRules framework

[1] https://nvkp.github.io/diploma-thesis-code/rulesets/jaur-yago/regionTotalLiftComputed.txt

was not utilized, because intended to be used in association rule mining for classification[23] rather than an explorative association rule mining, that this task aimed to be and I wanted the rules to be diverse by the content of the rules, not by the triples that instantiate the rules and to find rules with various chains of properties from the YAGO data set that can determine the measure value in the cube. After tuning the task's parameters and finding the *Golden mean* of the right number of discretizations, the algorithm found few rules that are worth mentioning. One of them is shown in the listing 8.1.

```
(?d <http://schema.org/memberOf> yago:FC_Nitra) ^ (?d <http://schema.org/birthPlace> ?
    c) ^ (?b <http://schema.org/containsPlace> ?c) ^ (?a czso:refArea ?b) ^ (?a qb:
    dataSet <jaur-districts-total>) -> (?a czso:
    dosazitelniNeumisteniUchazeciOZamestnani <[ 9639.0 ; 25767.0 ]_ef10_10/10>)

| support: 40, headCoverage: 0.01949317738791423, confidence: 0.6349206349206349,
    pcaConfidence: 0.6349206349206349, lift: 6.204081632653061, headConfidence:
    0.1023391812865497, headSize: 2052, bodySize: 63, pcaBodySize: 63
```

Listing 8.1: Rule relating to footballers

The rule states that in the time span covered by the cube, the districts in which a footballer was born, that has played for the Slovakian club PC Nitra, are associated with the highest number of available job applicants. The body size of the rule is 63 and the cube covers 9 years. It means that there are 7 districts with their natives, that even played for PC Nitra. Out of the 63 observations related to those districts, 40 contained the number of available job applicants in the highest of 10 intervals. Those districts were retrieved by a SPARQL query into the YAGO data set. The list contains districts with a heavy industry tradition, such as Most, Ostrava and Karviná and fairly populous districts such as Olomouc, Brno and Zlín. Both factors can explain the tendency to a higher number of job applicants since it is an absolute value.

Conclusions

The goal of this work was to explore the possibilities of enriching RDF data cube with the data from publicly available knowledge graphs and mining rules from this data. Section 6 contains a description of steps needed in the data pre-processing phase, a type of rules that can be mined from the data, and how this type affects the interest measures of the RDFRules algorithm based on how the input data is structured. This knowledge was used when performing the experiment described in section 7, which yielded results confirming the validity of the suggested approach, which was also a goal of this work. The RDFRules framework proved to be well suited for this type of experiment, thanks to its native recognition of the *owl* : *sameAs* predicates and ability to control the shape of the generated rule with its powerful rule pattern syntax. The source codes of the mining procedures and sets of the found rules are available in a Github repository at https://github.com/nvkp/diploma-thesis-code.

The extraction and pre-processing of the data were very time-consuming, but there is a lot of room to automate not only this task. For the dimension values of the examined cubes, automatic *record linkage* [21] tools could be used to automate the discovery of relevant triples in other data sources, which the data set can be augmented. Also, a tool can be imagined, that would examine the cubes' dimension value, which would infer the suitable way to cut the cube into the commensurable subcubes based on the hierarchy of the dimension value if they were described with SKOS or any other similar vocabulary, and on the characteristics of the measurements associated with the dimension values. Another tool could aid the post-processing phase by compensating for the interest measures' distortions based on the content of the rules and based on the structure of the cubes.

Regarding the RDFRules framework core API, as described in 6.4, the framework permits a definition of a rule pattern matching rules, that the algorithm cannot generate, which results in zero rules found. A functionality to the framework could be added, that would check for the validity of a rule pattern, and it would either try to repair the rule pattern by rearranging the atom patterns in the rule pattern's body or it would simply provide feedback to the user. Also, the user-friendliness of both the core API and the Web UI would be improved if not only single alphabetical characters moreover, in alphabetical order from the head, are allowed for the names of variables in the rule patterns.

List of References

[1] URL: https://www.w3.org/TR/swbp-skos-core-spec/.

[2] Rakesh Agrawal, Tomasz Imieli'nski, and Arun Swami. 'Mining association rules between sets of items in large databases
'. In: vol. 22. 1993-01, pp. 207–216. ISBN: 0897915925. DOI: 10.1145/170035.170072.

[3] David Beckett et al. *RDF 1.1 Turtle*. 2014-02. URL: http://www.w3.org/TR/turtle/.

[4] Tim Berners-Lee. *Linked Data - Design Issues*. 2006. URL: http://www.w3.org/DesignIssues/LinkedData.html.

[5] A. Berson and S.J. Smith. 'Data Warehousing, Data Mining, and OLAP'. In: (1997). ISSN: 00-7006-272-2.

[6] Richard Cyganiak and Dave Reynolds. *The RDF Data Cube Vocabulary*. W3C Recommendation. https://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/
. W3C, 2014-01.

[7] Luis Galárraga et al. 'Fast rule mining in ontological knowledge bases with AMIE+'. In: *The VLDB Journal* 24 (2015-07). DOI: 10.1007/s00778-015-0394-1.

[8] Luis Antonio Galárraga et al. 'AMIE: Association rule mining under incomplete evidence in ontological knowledge bases
'. In: 2013-05, pp. 413–422. DOI: 10.1145/2488388.2488425.

[9] P. Hájek, I. Havel, and M. Chytil. 'The GUHA method of automatic hypotheses determination'. In: *Computing* 1 (1966-12), pp. 293–308. DOI: 10.1007/BF02345483.

[10] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. 2001-01.

[11] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011. ISBN: 9781608454303. URL: http://linkeddatabook.com/.

[12] David CHUDÁN. 'Association rule mining as a support for OLAP [online]'. PhD thesis. Vysoká škola ekonomická v Praze, Praha, 2015 [cit. 2020-10-10]. URL: https://insis.vse.cz/zp/portal_zp.pl?prehled=vyhledavani;podrobnosti_zp=25910;zp=25910;download_prace=1.

[13] Benedikt Kämpgen and Andreas Harth. 'Transforming statistical linked data for use in OLAP systems'. In: 2011-09, pp. 33–40. DOI: 10.1145/2063518.2063523.

[14] Bohuslav Koukal. 'OLAP Recommender: Supporting Navigation in Data Cubes Using Association Rule Mining'. MSc. Thesis. Vysoká škola ekonomická v Praze, Praha, 2017.

[15] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. 'Fast and Exact Rule Mining with AMIE 3'. In: 2020-05, pp. 36–52. ISBN: 978-3-030-49460-5. DOI: 10.1007/978-3-030-49461-2_3.

[16] Frank Manola and Eric Miller. *RDF Primer*. 2004-02. URL: http://www.w3c.org/TR/rdf-primer/.

[17] Eric Prud'hommeaux and Andy Seaborne. *SPARQL Query Language for RDF*. 2008-01. URL: http://www.w3.org/TR/rdf-sparql-query/.

[18] Jan Rauch and Milan Šimůnek. 'Apriori and GUHA – Comparing two approaches to data mining with association rules'. In: *Intelligent Data Analysis* 21 (2017-08), pp. 981–1013. DOI: 10.3233/IDA-160069.

[19] *Recital 162 EU General Data Protection Regulation (EU-GDPR)*. 2020-05. URL: https://www.privacy-regulation.eu/en/recital-162-GDPR.htm.

[20] Capadisli Sarven, Sören Auer, and Reinhard Riedl. 'Towards Linked Statistical Data Analysis'. In: *CEUR (Central Europe workshop proceedings). Proceedings of the 1st International Workshop on Semantic Statistics* (2013). URL: http://csarven.ca/linked-statistical-data-analysis.

[21] Amit Sheth, Alfio Ferraram, and Andriy Nikolov. 'Data Linking for the Semantic Web'. In: 2013-01, pp. 169–200. DOI: 10.4018/978-1-4666-3610-1.ch008.

[22] Thomas Tanon, Gerhard Weikum, and Fabian Suchanek. 'YAGO 4: A Reason-able Knowledge Base'. In: 2020-05, pp. 583–596. ISBN: 978-3-030-49460-5. DOI: 10.1007/978-3-030-49461-2_34.

[23] Koen Vanhoof and Benoît Depaire. 'Structure of association rule classifiers: A review'. In: (2010-11). DOI: 10.1109/ISKE.2010.5680784.

[24] *YAGO: A High-Quality Knowledge Base*. URL: https://yago-knowledge.org/.

[25] Václav Zeman, Tomáš Kliegr, and Vojtěch Svátek. 'RDFRules Preview: Towards an Analytics Engine for Rule Mining in RDF Knowledge Graphs'. In: *RuleML Challenge* (2018).

[26] Václav Zeman, Tomáš Kliegr, and Vojtěch Svátek. 'RDFRules: Making RDF Rule Mining Easier and Even More Efficient'. In: *Semantic Web* 12 (2021). URL: http://www.semantic-web-journal.net/system/files/swj2511.pdf.

Appendices

A. SPARQL Queries

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX p: <http://www.wikidata.org/prop/>

CONSTRUCT {
    ?area p:P31 wd:Q5153359.
    ?area p:P6 ?headRole ;
        p:P131 ?superiorArea .
    ?headRole p:P6 ?person ;
        p:P580 ?headRoleStartDate ;
        p:P582 ?headRoleEndDate .
    ?person p:P102 ?partyRole .
    ?partyRole p:P580 ?partyRoleStartDate ;
            p:P582 ?partyRoleEndDate ;
            p:P102 ?party .
    ?party p:P1387 ?alignmentLabel .
}
WHERE {
    {
        ?area wdt:P31 wd:Q38911 .
    } UNION {
        ?area wdt:P31|p:P31 wd:Q5153359 .
        OPTIONAL {
            ?area wdt:P131|p:P131 ?superiorArea
        }
    } UNION {
        wd:Q3342946 wdt:P1269 ?area
    }
    OPTIONAL {
        ?area p:P6|wdt:P6 ?headRole .
        ?headRole ps:P6 ?person ;
            pqv:P580|pq:P580 ?headRoleStartDate .
        OPTIONAL {
            ?headRole pqv:P582|pq:P582 ?headRoleEndDate .
        }
        OPTIONAL {
            ?person p:P102|wdt:102 ?partyRole .
            OPTIONAL {
                ?partyRole pqv:P580|pq:P580 ?partyRoleStartDate .
            }
            OPTIONAL {
                ?partyRole pqv:P582|pq:P582 ?partyRoleEndDate .
            }
            OPTIONAL {
                ?partyRole ps:P102 ?party .
                ?party wdt:P1387|p:P1387 ?alignment .
                ?alignment rdfs:label ?alignmentLabel
                FILTER (lang(?alignmentLabel) = "en")
            }
        }
    }
}
```

Listing A.1: Extracting the Wikidata's political data about the Czech Republic

```
PREFIX cssa-ody:  <https://data.cssz.cz/ontology/days/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
```

```
PREFIX p: <http://www.wikidata.org/prop/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dt: <http://kizi.vse.cz/novp19/diploma-thesis/>

CONSTRUCT {
    ?headRole dt:appliesToRefPeriod ?refPeriod .
    ?partyRole dt:appliesToRefPeriod ?refPeriod .
}
WHERE {
    GRAPH <https://data.cssz.cz/resource/dataset/pomocne-ciselniky> {
        ?refPeriod skos:inScheme cssa-ody:DaysScheme .
        FILTER REGEX(str(?refPeriod), ".*year.*")
        FILTER REGEX(str(?refPeriod), "^.*\\d{4}-12-31")
        BIND(REPLACE(str(?refPeriod), "^.*(\\d{4}).*","$1") as ?refPeriodBind)
    }
    ?area p:P6 ?headRole .
    ?headRole p:P580 ?headRoleStartDate .
    FILTER ( datatype(?headRoleStartDate) = xsd:string)
    BIND(REPLACE(str(?headRoleStartDate), "^(\\d{4}).*","$1") as ?
        headRoleStartDateBind)
    OPTIONAL {
        ?headRole p:P582 ?headRoleEndDate .
        FILTER ( datatype(?headRoleEndDate) = xsd:string)
        BIND(REPLACE(str(?headRoleEndDate), "^(\\d{4}).*","$1") as ?
            headRoleEndDateBind)
    }
    FILTER (
        (bound(?headRoleEndDateBind) && ?refPeriodBind >= ?headRoleStartDateBind && ?
            refPeriodBind <= ?headRoleEndDateBind)
        ||
        (!bound(?headRoleEndDateBind) && ?refPeriodBind >= ?headRoleStartDateBind && ?
            headRoleStartDateBind > "2000")
    )
    OPTIONAL {
        ?headRole p:P6 ?person .
        ?person p:P102 ?partyRole .
        OPTIONAL {
            ?partyRole p:P580 ?partyRoleStartDate .
            FILTER (datatype(?partyRoleStartDate) = xsd:string)
            BIND(REPLACE(str(?partyRoleStartDate), "^(\\d{4}).*","$1") as ?
                partyRoleStartDateBind)
        }
        OPTIONAL {
            ?partyRole p:P582 ?partyRoleEndDate .
            FILTER ( datatype(?partyRoleEndDate) = xsd:string)
            BIND(REPLACE(str(?partyRoleEndDate), "^(\\d{4}).*","$1") as ?
                partyRoleEndDateBind)
        }
        FILTER (
            (bound(?partyRoleEndDateBind) && ?refPeriodBind >= ?partyRoleStartDateBind
                 && ?refPeriodBind <= ?partyRoleEndDateBind)
            ||
            (!bound(?partyRoleEndDateBind) && ?refPeriodBind >= ?
                partyRoleStartDateBind && ?partyRoleStartDateBind > "2000")
            ||
            (!bound(?partyRoleStartDateBind) && !bound(?partyRoleEndDateBind))
        )
    }
}
```

Listing A.2: Creating the **appliesToRefPeriod** triples