Mining Semantic Association Rules from RDF Data

Mayuri Mendke[1], Amrit Kaur[1] and Pratik Kumar Agarwal[1]
Supervisors: Dr. Hajira Jabeen, Gezim Sejidu

Informatik II, Universität Bonn, Germany
s6mamend@uni-bonn.de, s6amkaur@uni-bonn.de, s6pragar@uni-bonn.de

Abstract

This paper focuses on Association rule mining, which is one of the most effective techniques for detecting frequent itemsets from the given dataset and generate rules from them by considering knowledge at both instance and schema level. The dataset is a Semantic Web data where information is represented in the form of RDF triples (Subject, Predicate, Object). For mining rules we introduced an approach called SWARM (Semantic Web Association Rule Mining) which reveals common behavioural patterns associated with knowledge at the instance-level and schema-level. Results show that utilising knowledge at schema level is essential for appropriate rule mining.

1 Introduction

The Semantic Web (SW) is an effort to make knowledge on the Web both human-understandable and machine-readable. As the SW data are represented in the form of RDF triples and are huge, the knowledge bases (KBs) which is formed by the inter-connections of different triples, suffer from issues like incompleteness and incorrectness. In this regard, association rule mining helps in discovering frequent patterns from the dataset.

Triples	Subject	Predicate	Object
T1	John Lennon	Instrument	Guitar
T2	John Lennon	Occupation	Songwriter
T3	George Harrison	Instrument	Guitar
T4	George Harrison	Occupation	Songwriter
T5	John Lennon	rdf:type	Person
T6	George Harrison	rdf:type	Musical Artist
T7	Musical Artist	rdf:subClassOf	Person

Table 1: Some RDF triples from the dataset

Generated Rule: {Person} : (instrument, Guitar) $\Rightarrow$ $(occupation, Songwriter)$

The above rule shows that most of the time people who play a musical instrument (e.g., guitar) are probably songwriters. The goal of this paper is to generate such behavioural patterns from KBs. Mining such regularities help us gain better understanding of SW data.
We design an approach called SWARM (Semantic Web Association Rule Mining) which can automatically mine and generate semantically-enriched rules and exploit knowledge encoded at the schema-level (rdf:type and rdfs:subClassOf relations) along with knowledge at instance level, to enrich semantics of rules similar to the "Generated Rule" from Table 1 shown above. Unlike

earlier approaches which focuses only on the information at instance level, we demonstrate that ignoring knowledge encoded at the schema-level negatively impacts the interpretation of discovered rules. The quality of the rule is measured by Support, Confidence, and Lift parameters.

2 Approach

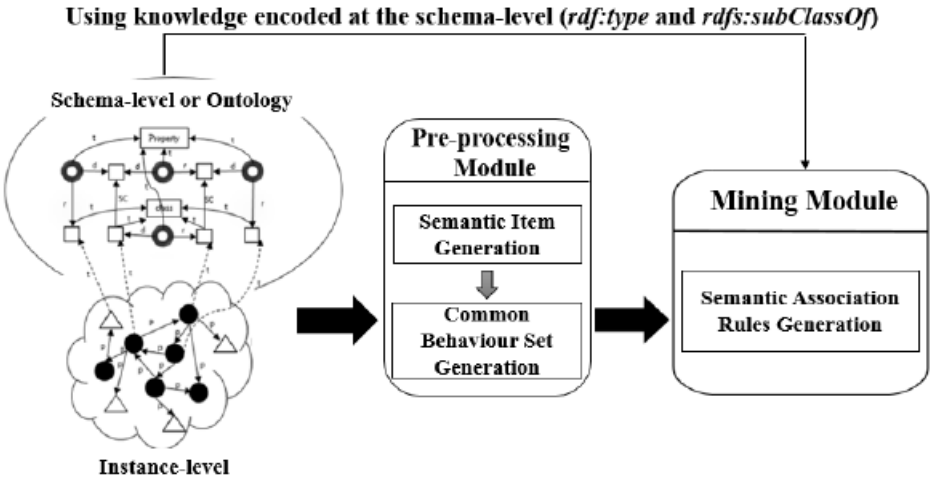

Figure 1: SWARM Model

The model we plan to use in this project is shown in Figure 1. Two major modules of this model are Pre-Processing and Mining. The Pre-Processing module will help in processing our RDF triples and the Mining Module as the name suggests will help in generating semantic association rules. And finally our model will evaluate the quality of the generated rules by using rdf:type and rdfs:subClassOf relations in the ontology.

As our main goal is to extract behavioural pattern among entities, pre-processing module is divided into two sub-modules: Semantic Item Generation(Algorithm 1) and Common Behaviour Set Generation(Algorithm 2) to show common behaviours of entities.
Using Resilient Distributed Datasets (RDD) data structures, help us in successful implementation of these modules. At the core, an RDD is an immutable distributed collection of elements of our data, partitioned across nodes in cluster that can be operated in parallel with a low-level API that offers transformations and actions. The DBpedia dataset is used in this project.

Algorithm 1: Semantic Item Generation

input : $Triples$
output: SI

$SI \leftarrow \emptyset$;
$foundFlag \leftarrow false$;
for *each triple* $t_j \in Triples$ **do**
  $pa' \leftarrow$ the Pair of t_j
  for *each* $si_i \in SI$ **do**
    if $pa' = si_i.pa$ **then**
      add the subject of t_j to $si_i.es$;
      $foundFlag \leftarrow true$;
  if $foundFlag = false$ **then**
    new Semantic Item si'_i;
    $si'_i \leftarrow t_j$;
    add si'_i to SI;
  $foundFlag \leftarrow false$;
return SI

Algorithm 2: Total Common Behavior Set Generation

input : SI, $SimTh$
output: $TCBS$

$TCBS \leftarrow \emptyset$;
$foundFlag \leftarrow false$;
for *each* $si_i \in SI$ **do**
  for *each* $cbs_j \in TCBS$ **do**
    $set_a \leftarrow \bigcap_{si_i \in cbs_j} si_i.es$;
    $set_b \leftarrow \bigcup_{si_i \in cbs_j} si_i.es$;
    $SD \leftarrow |si_i.es \cap set_a| / |si_i.es \cup set_b|$;
    if $SD \geq SimTh$ **then**
      add si_i to cbs_j ;
      $foundFlag \leftarrow true$;
  if $foundFlag = false$ **then**
    new Common Behavior Set cbs'_j;
    $cbs'_j \leftarrow si_i$;
    add cbs'_j to $TCBS$;
  $foundFlag \leftarrow false$;
return $TCBS$

3 Implementation

3.1 Pre-processing Module

3.1.1 Semantic Item Generation

Consider the Example of Table 2. The generated example is the expected output which is generated by the Semantic Item Generation sub-module, given our dataset in Table 1 as input.

	Semantic Items
si_1	$\{JohnLennon, GeorgeHarrison\}(Instrument, Guitar)$
si_2	$\{JohnLennon, GeorgeHarrison\}(Occupation, Songwriter)$

Table 2: Example of Semantic Items

Now, let's see what this generated example is telling. We can now define Semantic Item as:

Semantic Item: A Semantic Item si_j is a 2-tuple. The first part contains a list of subjects or objects and the second part contains a pair of predicate-subject or predicate-object.

This shows that there is a common behaviour which is shared by certain number of entities. Like in this case John and George shares a common behaviour of playing Guitar. This module will generate all such pair present in our knowledge base. If there is no existing Semantic Item that contains a particular Pair, then the algorithm creates a new Semantic Item $si_j^{'}$ and adds it to SI.

3.1.2 Common Behaviour Set Generation

Consider the Example of Table 3. The generated set is the expected output which is generated by the Common Behaviour Set Generation Module, given Semantic Items and a threshold value called similarity threshold (SimTH) as input.

	Common Behaviour Sets
cbs_1	$\{JohnLennon, GeorgeHarrison\}(Instrument, Guitar)$
	{John Lennon, George Harrison} (Occupation, Songwriter)

Table 3: Example of Common Behaviour Sets

From the generated set in Table 3 we can easily identify that this module is combining all the common behaviours or activities taken by similar group of entities into a single set. Items can be aggregated into the same *cbs*, if the similarity degree (SD) of their Element Sets are greater than or equal to Similarity Threshold (SimTh). SD of element sets (es_j) can be calculated as:

$$SD(es_a, es_b, ..., es_c) = \frac{|es_a \cap es_b \cap ..... \cap es_m|}{|es_a \cup es_b \cup ..... \cup es_m|}$$

The output of this sub-module will be a Total Common Behavior Set (TCBS) = {cbs1, cbs2, ..., cbsn} that contains all Common Behavior Sets. For each semantic item si_j SD of Element Sets of si_j is calculated and stores the result in SD. If the SD is greater than or equal to SimTh, then the algorithm adds si_j to cbs_j. If there is no existing Common Behaviour Set with a similar Element Set, then the algorithm creates a new cbs, i.e., $cbs_j^{'}$ and adds it to TCBS.

3.2 Mining Module

Association rule mining is fundamentally focused on discovering frequent co-occurring associations among a collection of items. Given a set of transactions, association rule mining aims to discover the rules to predict the occurrence of a particular item. From 3.1.2 we have Common Behavior Sets(cbs), taken as transactions and this will be an input for mining module to generate Semantic Association Rules in the context of semantic web data. In order to see the significance of semantics in the generated rules, quality factors (Support, Confidence and Lift) are defined by using knowledge both at instance-level and schema-level.

3.2.1 Semantic Association Rules Generation

From given cbs_j, The Semantic Association Rule r is composed of two parts:

Antecedent pairs: $pa_{ant} = \{pa_1, ..., pa_n\}$ (Some pairs from cbs_j)
Consequent pairs: $pa_{con} = \{pa_{n+1}, ..., pa_m\}$ (Remaining pairs from cbs_j)

Rule r holds a common Rules Element Set res which is the union of Element Sets in the cbs_j, i.e., $\{es_1 \cup ... \cup \text{es}_m\}$.

$$res : pa_{ant} \Rightarrow pa_{con} \tag{1}$$

Here res is a common Rules Element Set containing $\bigcup_{si_i \epsilon cbs_j} si_i.es$, pa_{ant}, pa_{con} ϵcbs$_j$ and $pa_{ant} \cap \text{pa}_{con} = \emptyset$.

Table 4 shows an example of rule generated at instance level from Common Behavior Sets in Table 3.

	Semantic Association Rule
r_1	{John Lennon, George Harrison}:(Instrument, Guitar) $\Rightarrow$ $(Occupation, Songwriter)$

Table 4: Example of Semantic Association Rule at instance level

Since we are majorly focusing on rules generated at schema level, hence we will consider schema relations (rdf:type and rdfs:subClassOf) for the element sets of our instance level rules. Here, John Lennon has rdf:type Person and George Harrison has rdf:type MusicalArtist. Since, MusicalArtist is rdfs:subClassOf Person, then the instances of MusicalArtist class also belong to the Person class. therefore, the common class among both the instances is class Person, so we'll replace instances with class Person and rule will be generated at the semantic level. Here pair of antecedent and consequent remains same.
Table 5 shows an example of rule generated at schema level from Common Behavior Sets in Table 3.

	Semantic Association Rule
r_1	{Person}:(Instrument, Guitar) $\Rightarrow$ $(Occupation, Songwriter)$

Table 5: Example of Semantic Association Rule at Schema level

4 Evaluation

The generated rules are further evaluated on the basis of quality factors in order to find out interestingness of newly discovered rules. Three quality factors; Support, Confidence, and Lift are introduced to evaluate rules generated in 3.2.1 that consider knowledge not only at the instance-level but also at the schema-level.

Support. Consider a Semantic Association Rule r, the Support sup(r) can be calculated by using Equation 2:

$$sup(r) = \frac{|\bigcup_{ins_j \epsilon c_i \wedge ins_j \epsilon res_k \wedge ins_j.pa_{ant_k}} c_i|}{|\bigcup_{ins_j \epsilon c_i \wedge ins_j \epsilon res_k} c_i|} = \frac{|Person \cap \text{instrument Guitar}|}{|Person|} \tag{2}$$

where $ins_j \epsilon c_i$ is an instance of Class c_i, $ins_j \epsilon \text{res}_k$ is an instance of Rules Element Set res_k, and $ins_j.pa_{ant_k}$ shows those instances that hold pa_{ant_k} as the Pairs. Here r is of the form Equation 1.

For example, support value for rule generated in table 4 will be, sup= 0.04. Here, there are two instances which share instrumentGuitar as a Pair which is the numerator value of given fraction. The total number of instances of Person class which is six is the denominator value of given fraction.

Confidence. Consider a Semantic Association Rule r, the Confidence conf(r) can be calculated by using Equation 3:

$$conf(r) = \frac{|\bigcup_{ins_j \epsilon c_i \wedge ins_j \epsilon res_k \wedge ins_j.pa_{ant_k} \wedge ins_j.pa_{con_k}} c_i|}{|\bigcup_{ins_j \epsilon c_i \wedge ins_j \epsilon res_k \wedge ins_j.pa_{ant_k}} c_i|} \tag{3}$$

$$\text{conf(r1)} = \frac{|Person \cap \text{instrument Guitar} \cap \text{occupation Songwriter}|}{|Person|}$$

where ins_j ϵc_i is an instance of Class c_i, ins_j ϵres_k is an instance of Rules Element Set res_k, and $ins_j.pa_{ant_k}$ and $ins_j.pa_{con_k}$ show those instances that hold pa_{ant_k} and pa_{con_k} as the Pairs, respectively. Here r is of the form Equation 1.
The confidence value for rule generated in table 4 will be, conf= 1.0. Here, the numerator is the total number of instances belonging to the Person class that contains instrumentGuitar and occupationSongwriter as the Pairs. The total number of instances of Person class which is six is the denominator value of given fraction.

Lift. Consider a Semantic Association Rule r, the lift(r) can be calculated by using Equation 4:

$$lift(r) = \frac{a}{b \times c} \tag{4}$$

where,

$$a = |\bigcup_{ins_j \epsilon c_i \wedge ins_j \epsilon res_k \wedge ins_j.pa_{ant_k} \wedge ins_j.pa_{con_k}} c_i|,$$

$$b = |\bigcup_{ins_j \epsilon c_i \wedge ins_j \epsilon res_k \wedge ins_j.pa_{ant_k}} c_i|,$$

$$c = |\bigcup_{ins_j \epsilon c_i \wedge ins_j \epsilon res_k \wedge ins_j.pa_{con_k}} c_i|$$

Here r is of the form Equation 1.
Lift is a quality factor for the deviation of a rule from the model of statistic independence of the antecedent and consequent. The lift of an association rule measures the interestingness of a rule. If a lift ratio is greater than 1, it indicates that the occurrence of the antecedent has a positive effect on the occurrence of the consequent. If a lift value is smaller than 1, it indicates that the antecedent and the consequent appear less often together and the occurrence of the antecedent has a negative effect on the occurrence of the consequent. If a lift value is about 1, it indicates that the occurrence of the antecedent has almost no effect on the occurrence of the consequent. Therefore, the larger the lift value, the more significant the association.

We tested SWARM on this dataset to see how readable the generated rules are. For each sample dataset, we conducted three separate experiments by applying 60%, 70%, and 80% minimum Similarity Thresholds.

Table 6 represents some Semantic Association Rules of Person class generated by SimTh=60%. In this table, at least 60% of instances of Element Sets satisfy the rules. Table 7 shows some rules generated by SimTh = 70%. Table 8 shows some rules generated by SimTh = 80%.Table 9 shows the average value of Support, Confidence,and Lift of the generated rules with different Similarity Thresholds. One can observe that SimTh has a direct effect on the number of generated rules. Obviously, by increasing the SimTh, the number of generated rules will be decreased.

Rules	**Rules Element Set**	**Semantic Association Rule**	**sup.**	**conf.**	**lift**
r_1	{Ayn Rand, Albert Camus}	{Person}:(influencedBy, FriedrichNietzsche) $\Rightarrow (influencedBy, FyodorDostoyevsky)$	0.03	1.0	33.33

Table 6: Examples of Semantic Association Rules discovered from Person class (SimTh=60%)

Rules	**Rules Element Set**	**Semantic Association Rule**	**sup.**	**conf.**	**lift**
r_1	{Apollo 14, Apollo 17}	{Event}:(booster, Saturn V) $\Rightarrow$ $(launchPad, KennedySpaceCenterLaunchComplex39)$	0.27	1.0	3.7

Table 7: Examples of Semantic Association Rules discovered from Person class (SimTh=70%)

Rules	**Rules Element Set**	**Semantic Association Rule**	**sup.**	**conf.**	**lift**
r_1	{John Lennon, George Harrison}	{Person}:(Instrument, Guitar) $\Rightarrow (Occupation, Songwriter)$	0.04	1.0	25.0

Table 8: Examples of Semantic Association Rules discovered from Person class (SimTh=80%)

Class	Min SimTh%	Avg sup.	Avg conf.	Avg lift
Person	80	0.13	1.0	9.73
	70	0.13	1.0	9.73
	60	0.11	0.94	12.08
Organization	80	0.15	1.0	6.67
	70	0.15	1.0	6.67
	60	0.15	1.0	6.67
Work	80	0.07	1.0	17.25
	70	0.07	1.0	17.25
	60	0.07	1.0	17.25
Event	80	0.81	1.0	1.89
	70	0.61	1.0	2.78
	60	0.61	1.0	2.78

Table 9: The quality measures of generated rules from different classes of DBpedia ontology

5 Project Timeline

Table 10 shows how tasks were discussed, defined, assigned and executed.

Week	Task Committed
1	Thorough Reading Going Through Spark/Scala Tutorial
2	Discussing Algorithm With Teammates Planning And Focusing Major Modules Dividing Modules
3	Discussion With Supervisors Analyzing our approach Analyzing Supervisor's suggestions
4	Collecting Data Set Implementing First Algorithm Of Pre-processing Module
5	Implementing Second Algorithm Of Pre-processing Module Implementing Mining Module: Rule forming at instance level
6	Implementing Rule Formation at schema level Calculating Support, Confidence and Lift Values
7	Evaluating Overall Performance Optimizing Code
8	Giving Final Modification To Code(adding comments etc.) Writing Report

Table 10: Project Timeline

6 Future Work and Improvement

Here we considered rdf:type and rdfs:subClassOf relations at the schema-level which generates semantically-enriched rules. But in future, we will further look into other semantic relations (e.g., rdfs:subPropertyOf and rdf:Property)in order to generate more enriched association rules. As future work, we would like to develop an approach to remedy conflicts such as lack of correctness and inconsistency between entities at the instance-level and corresponding

class information in the ontology. Another important direction for future work is to apply some constraints on the Rules Element Sets to learn new knowledge from the discovered rules.

7 Setting up the environment

To run this project, Apache spark and Scala must be installed in system or install Scala IDE(Eclipse).

1. Download and generate project with all its dependencies using Maven template from SANSA Template Maven Spark.
2. Open this project as Maven project in Scala IDE.
   (a) Click on File then Import
   (b) Select Maven then Existing Maven project
   (c) Select the downloaded project which contains .pom file by browsing to that project directory.
3. To use Scala 2.12 bundle configure Scala compiler.
4. Right click on project and select Properties.
5. Check "Use Project Settings" and select 2.12 bundle at Scala compiler for scala installation.
6. Click at "Apply and close" and run the program.

For this project execution, no additional libraries are required.

References

1. Molood Barati, Quan Bai, Qing Liu. *Mining Semantic Association Rules from RDF Data* S0950-7051(17)30325-8, KNOSYS 3973, 2017.
2. https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html
3. https://github.com/SANSA-Stack/SANSA-OWL/tree/develop/sansa-owl-common