Mining RDF Data for Property Axioms

Sergey Kuptsov[1]
[1] Universität Bonn, Germany
s6sekupt@uni-bonn.de

Abstract

The presented report deals with the extraction of the Property Axioms from the RDF data based on Association Rule Mining. Lines of RDF are analyzed and transformed in a certain Transaction Database for certain Property Axioms (e.g. Object Property Symmetry Axiom, Object Property Subsumption or Disjointness Axioms etc.). In order to extract Object Property Axioms Association Rules of certain support and confidence are generated from obtained Transaction Database. In this Lab the FP-Growth Algorithm ["*An implementation of the FP-Growth Algorithm", Christian Borgelt, 2010*] is used for Association Rule Mining. A Workflow starting From RDF reading using SANSA Stack [http://sansa-stack.net/], generating Transaction Databases for certain Object Property Axioms, Association Rule Mining with FP-Growth and final Property Axioms extraction was implemented.

1 Introduction

1.1. Problem Definition

Growing amount of linked Data on the WEB in sense of coverage and size is a very good basis for many interesting applications, in particular deriving of an implicit informations and verification of the data. In case of deriving implicit information extracting Property Axioms from the Linked data is very important for completing the schema of the data. For example if the information about the actors and movies are contained in the RDF Graph, but the link "*Terrens Howarth" actedIn "Iron Man"* is stated in the RDF, the query for all actors of the "Iron Man" might yield incomplete results, unless the two properties *starring* and *actedIn* are declared as inverse Object Properties. Having this implicit knowledge will add the missing link in the RDF Graph yielding correct answers for the corresponding queries.

Thus methods for mining RDF Data for Property Axioms are required and are developed. One of the reasonable approaches is to use Association Rule Mining based approaches[David,J. et. al. "*Association rule onthology matching approach*" 2007; Mädche,A. and Staab,S., "*Discovering conceptual relations from text*", 2000 etc.]. **Association rule mining** is a procedure which is meant to find frequent patterns, correlations, associations, or causal structures from data sets found in various kinds of databases such as relational databases, transactional databases, and other forms of data repositories [https://www.techopedia.com/definition/30306/association-rule-mining]. Applying that to the large RDF Graph we may obtain knowledge about the links between nodes which are the Properties connecting Objects and Subjects. Object, Predicate and Subject form the so called RDF-Triples and are a good basis for Transaction Database generation used for Association Rules generation. Taking an association rule of a high confidence level, we may extract an Object Property Axiom related to a particular predicate.

1.2. Association Rule Mining.

An introduction to the notion of Association Rule Mining is necessary in order to follow further explanations. Initially Association Rule Mining was developed for analyzing shopping carts of huge shops for products commonly bought together. That knowledge would enable better recommendations. Due to this specific nature association rule mining algorithms are developed for huge and sparse datasets, thus are a god choice for the data found in the large Linked Datasets.

Formally, association rule mining acts on the transaction database $D=(t_1,t_2, \dots , t_n)$ containing transactions over a set I of all possible items. In applications often this database D is represented as so called transaction table where each line has a transaction ID and is represented as a sequence of 0 and 1. 1 in i^{th} position means the transaction has the item i from I, 0 means item i is not present in the transaction.

In order to generate an association rule it is first necessary to generate all frequent items. There is a few Algorithms for this purposes, one of them is an FP-Growth algorithm, where FP stays for Frequent Pattern. In the first pass, the algorithm counts occurrence of items (attribute-value pairs) in the dataset, and stores them to 'header table'. In the second pass, it builds the FP-tree structure by inserting instances. Items in each instance have to be sorted by descending order of their frequency in the dataset, so that the tree can be processed quickly. Items in each instance that do not meet minimum coverage threshold are discarded. If many instances share most frequent items, FP-tree provides high compression close to tree root. Recursive processing of this compressed version of main dataset grows large item sets directly, instead of generating candidate items and testing them against the entire database. Growth starts from the bottom of the header table (having longest branches), by finding all instances matching given condition. New tree is created, with counts projected from the original tree corresponding to the set of instances that are conditional on the attribute, with each node getting sum of its children counts. Recursive growth ends when no individual items conditional on the attribute meet minimum support threshold, and processing continues on the remaining header items of the original FP-tree. Once the recursive process has completed, all large item sets with minimum coverage have been found, and association rule creation begins[Witten, Frank, Hall: Data mining practical machine learning tools and techniques, 3rd edition].

Frequent items are items are the ones from $X \subset I$ that exceed some certain support value in the given transaction database D. This support value $supp(X)$ is the number of transactions from D containing items from X. Based on these items the association rules of the form $A \rightarrow B$ are generated. Confidence value is a quality measure of certain association rule regarding a specific transaction database, $conf(A \rightarrow B)=supp(A \cup B)/supp(A)$. The latter describes the ratio of transaction for which the rule is valid to the number of transactions it is applicable to. Manipulating the confidence value enables generating association rules of certain guaranteed quality with respect to a specific dataset.

Problem Definition: *Given a specific RDF input extract and list certain Object Property Axioms.*

2 Approach

In this project we follow the Association Rule Mining strategy in order to extract Object Property Axioms from a given RDF Data. We first translate parsed RDF into a suitable transaction database, which is after used as the input to the FP-Growth algorithm. We obtain a set of

association rules, which are afterwards translated into the Object Property Axioms. The above workflow is broadly described and presented in [Fleischhacker, D., Völker, J., Stuckenschmidt, H. "*Mining RDF Data for Property Axioms*"]. In this report only a part of the properties described in the latter paper is investigated.

In order to illustrate workflow described above we describe a case of Object Property Symmetry. Starting by reading and parsing RDF-triples we build a foundation for the future transaction database. To generate symmetry axioms we acquire all possible pairs of Objects and Subjects connected via Predicates in the RDF-triples, this pairs are now unique Transactions ID. In order to form a suitable transaction in transaction database for symmetry property axioms we need to introduce an inverse property identifier. Formally, building a transaction for a unique Object-Subject pair we add a direct property identifier r to the transaction as well as an inverse property $r^{\wedge}$, if this property connects given Object and Subject in an RDF-triple.
After we parse association rules extracted from the formed transaction database for specific patterns, namely $r \rightarrow r^{\wedge}$, where the first identifier is the normal property identifier, whereas the second is the inverse property identifier for the same property. The association rules fitting the pattern described above exceed certain user specified confidence threshold induces an Object Property Symmetry Axiom. In the above case a corresponding symmetry axiom is *Sym(r).*

Further we explain how to build a suitable transaction database in order to extract Subsumption and Disjointness Property Axioms from a given RDF-dataset. In order to form a transaction for the Subsumption and Disjointness Axiom mining we need to introduce a *complement-identifier* not_r for a direct identifier r of a property. Each transaction contains a complement-identifier of a property if and only if it does not contain a corresponding direct identifier of a property. In this transaction dataset we will be looking for two patterns, namely $r_i \rightarrow not\_r_j$ for Disjointness property and $r_i \rightarrow r_j$ for Subsumption property.

3 Implementation

For the implementation of the described above Workflow Scala programming language and Spark framework were used. For Parsing RDF input data SANSA Stack is used. In order to generate Transaction Database authors own methods were implemented:

`extractTransactionsSymmetryAxiom(df:DataFrame)`: takes predicates corresponding to the unique Object-Subject pair and adds direct and inverse property identifiers to the transaction.

`extractTransactionsSubsumptionAndDisjointnessAxiom(df:DataFrame,predicates: DataFrame)`: takes predicates corresponding to the unique Object-Subject pair and adds direct and complement property identifiers as described in Section 2 to the transaction.

For the extractions of the corresponding Object Property Axioms two methods were implemented:

`extractSymmetryAxiom(df :DataFrame)`: Searches for the patterns in the generated association rules and generates Symmetry property axioms.

`extractSubsumptionDisjointnessAxiom(df :DataFrame)`: Searches for the patterns in the generated association rules and generates Disjointness or Subsumption property axioms based on the found pattern.

For the Association Rules generation Sparks FP-Growth method of the Machine Learning library was used.[https://spark.apache.org/docs/2.2.0/ml-frequent-pattern-mining.html].

4 Evaluations

For the purpose of demonstration of the capability of the implemented workflow a small and transparent example of RDF was used:

<a><hates><c>.
<a><friendOf><b>.
<a><descendant><b>.
<b><ancestor><a>.
<b><friendOf><a>.
<c><ancestor><b>.
<b><descendant><c>.
<c><hates><a>.
<c><friendOf><b>.
<b><friendOf><c>.
<b><friendOf><d>.
<f><friendOf><c>.
<a><sees><b>.
<b><sees><a>.
<a><sees><c>.
<c><sees><a>.
<b><sees><c>.
<c><sees><b>.

We can clearly see the types of the object properties presented in the above example. But first let see the example of Symmetry axioms extraction run of the implemented workflow.

```
===========Input DataFrames===========
+---+----------+---+
|  s|         p|  o|
+---+----------+---+
|  a|     hates|  c|
|  a|  friendOf|  b|
|  a|descendant|  b|
|  b|  ancestor|  a|
|  b|  friendOf|  a|
|  c|  ancestor|  b|
|  b|descendant|  c|
|  c|     hates|  a|
|  c|  friendOf|  b|
|  b|  friendOf|  c|
|  b|  friendOf|  d|
```

```
|  f|  friendOf|  c|
|  a|      sees|  b|
|  b|      sees|  a|
|  a|      sees|  c|
|  c|      sees|  a|
|  b|      sees|  c|
|  c|      sees|  b|
+---+----------+---+

===========Predicates===========
+----------+
|         p|
+----------+
|      sees|
|descendant|
|  friendOf|
|     hates|
|  ancestor|
+----------+

===========Transactions Extracted Symmetry===========
friendOf friendOf^
ancestor ancestor^ friendOf friendOf^ sees sees^
hates hates^ sees sees^
friendOf friendOf^
descendant descendant^ friendOf friendOf^ sees sees^
ancestor ancestor^ friendOf friendOf^ sees sees^
hates hates^ sees sees^
friendOf friendOf^ descendant descendant^ sees sees^
```

Above is the result of the Transaction database generation. We see that for each of the presented Object-Subject pair a transaction containing respective direct and inverse property identifiers were added.

```
===========Frequent Items===========
+--------------------+----+
|               items|freq|
+--------------------+----+
|              [sees]|   6|
|          [friendOf]|   6|
|    [friendOf, sees]|   4|
|         [friendOf^]|   6|
|[friendOf^, frien...|   6|
|[friendOf^, frien...|   4|
|   [friendOf^, sees]|   4|
|             [sees^]|   6|
|  [sees^, friendOf^]|   4|
|[sees^, friendOf^...|   4|
|[sees^, friendOf^...|   4|
```

```
|[sees^, friendOf^...|   4|
|   [sees^, friendOf]|   4|
|[sees^, friendOf,...|   4|
|       [sees^, sees]|   6|
+--------------------+----+
```

Among the items the frequent ones were found in order to start association rule generation.

```
===========Association Rules Symmetry==========
+--------------------+-----------+----------+
|          antecedent| consequent|confidence|
+--------------------+-----------+----------+
|   [friendOf^, sees]| [friendOf]|       1.0|
|   [friendOf^, sees]|    [sees^]|       1.0|
|             [sees^]|     [sees]|       1.0|
|[sees^, friendOf^...|     [sees]|       1.0|
|    [friendOf, sees]|[friendOf^]|       1.0|
|    [friendOf, sees]|    [sees^]|       1.0|
|[sees^, friendOf^...| [friendOf]|       1.0|
|   [sees^, friendOf]|[friendOf^]|       1.0|
|   [sees^, friendOf]|     [sees]|       1.0|
|              [sees]|    [sees^]|       1.0|
|         [friendOf^]| [friendOf]|       1.0|
|[sees^, friendOf,...|[friendOf^]|       1.0|
|  [sees^, friendOf^]| [friendOf]|       1.0|
|  [sees^, friendOf^]|     [sees]|       1.0|
|[friendOf^, frien...|    [sees^]|       1.0|
|          [friendOf]|[friendOf^]|       1.0|
+--------------------+-----------+----------+
```

Above we see the table of the association rules together with the confidence levels.

```
Sym(sees)
Sym(friendOf)
```

As the result of the Axiom extraction method we see that two predicates “sees” and “friendOf” were found to be symmetric which was expected.

Next we see the result of work for Disjointness and Subsumption properties for the same input RDF.

```
===========Transactions Extracted Subsumption and Disj==========
friendOf not_sees not_descendant not_hates not_ancestor
ancestor friendOf sees not_descendant not_hates
hates sees not_descendant not_friendOf not_ancestor
friendOf not_sees not_descendant not_hates not_ancestor
descendant friendOf sees not_hates not_ancestor
ancestor friendOf sees not_descendant not_hates
hates sees not_descendant not_friendOf not_ancestor
friendOf descendant sees not_hates not_ancestor
```

Above is the result of the Transaction database generation. We see that for each of the presented Object-Subject pair a transaction containing respective direct and complement property identifiers were added.

```
==========Frequent  Subsumption and Disj==========
+--------------------+----+
|               items|freq|
+--------------------+----+
|      [not_ancestor]|   6|
|[not_ancestor, no...|   4|
|[not_ancestor, fr...|   4|
|[not_ancestor, no...|   4|
|[not_ancestor, no...|   4|
|[not_ancestor, sees]|   4|
|              [sees]|   6|
|          [friendOf]|   6|
|    [friendOf, sees]|   4|
|    [not_descendant]|   6|
|[not_descendant, ...|   4|
|[not_descendant, ...|   4|
|         [not_hates]|   6|
|[not_hates, not_d...|   4|
|[not_hates, not_d...|   4|
|[not_hates, frien...|   6|
|[not_hates, frien...|   4|
|   [not_hates, sees]|   4|
+--------------------+----+
```

Among the items the frequent ones were found in order to start association rule generation.

```
==========Association Rules  Subsumption and Disj==========
+--------------------+-----------+----------+
|          antecedent| consequent|confidence|
+--------------------+-----------+----------+
|[not_hates, not_d...| [friendOf]|       1.0|
|[not_ancestor, no...| [friendOf]|       1.0|
|         [not_hates]| [friendOf]|       1.0|
|    [friendOf, sees]|[not_hates]|       1.0|
|   [not_hates, sees]| [friendOf]|       1.0|
|[not_ancestor, fr...|[not_hates]|       1.0|
|[not_descendant, ...|[not_hates]|       1.0|
|          [friendOf]|[not_hates]|       1.0|
+--------------------+-----------+----------+

Subsumes(not_hates,friendOf)
Disj(friendOf,hates)
```

As the result of the Axiom extraction method we see reasonable results. For purpose of demonstration of the capabilities of the method implemented a Subsumption Property was artificially added to the output, since the initial input data does not have an explicit candidate for that.

5 Project timeline

Authors: Sergey Kuptsov

Date	*Planned work*	*Implementor*
12.06.2018-19.06.2018	Study of literature, finding references, investigating framework	Sergey kuptsov
20.06.2018-28.06.2018	Study of literature, implementation of input reading and parsing. Implementing of the Transactiondatabase generating methods.	Sergey Kuptsov
28.06.2018-9.07.2018	Debugging, code improvement, workflow corrections	Sergey Kuptsov
3.08.2018-7.08.2018	Testing the program. Debugging. Writing report.	Sergey Kuptsov
28.08.2018-29.08.2018	Report checking. Final preparations for the project presentation.	Sergey kuptsov

References

Christian Borgelt, “An implementation of the FP-Growth Algorithm”, 2010.

http://sansa-stack.net/

David,J. et. al. “Association rule onthology matching approach” 2007.

Mädche,A. and Staab,S., “Discovering conceptual relations from text”, 2000.

https://www.techopedia.com/definition/30306/association-rule-mining

Witten, Frank, Hall: Data mining practical machine learning tools and techniques, 3rd edition.

Fleischhacker, D., Völker, J., Stuckenschmidt, H. “Mining RDF Data for Property Axioms.

https://spark.apache.org/docs/2.2.0/ml-frequent-pattern-mining.html