Datenbank Spektrum (2013) 13:111–120
DOI 10.1007/s13222-013-0126-x

SCHWERPUNKTBEITRAG

Improving RDF Data Through Association Rule Mining

Ziawasch Abedjan · Felix Naumann

Received: 23 January 2013 / Accepted: 25 April 2013 / Published online: 21 May 2013

Abstract Linked Open Data comprises very many and often large public data sets, which are mostly presented in the RDF triple structure of subject, predicate, and object. However, the heterogeneity of available open data requires significant integration steps before it can be used in applications. A promising and novel technique to explore such data is the use of association rule mining. We introduce "mining configurations", which allow us to mine RDF data sets in various ways. Different configurations enable us to identify schema and value dependencies that in combination result in interesting use cases. We present rule-based approaches for predicate suggestion, data enrichment, ontology improvement, and query relaxation. On the one hand we prevent inconsistencies in the data through predicate suggestion, enrichment with missing facts, and alignment of the corresponding ontology. On the other hand we support users to handle inconsistencies during query formulation through predicate expansion techniques. Based on these approaches, we show that association rule mining benefits the integration and usability of RDF data.

Keywords Linked open data · RDF · Association rule mining · Schema analysis

1 Inconsistency in RDF Data

The increasing amount of Linked Open Data (LOD) on the Web raises new opportunities and challenges for the data mining community [14]. LOD is often represented in the Resource Description Framework (RDF) data model: a triple structure consisting of a subject, a predicate, and an object (SPO). Each triple represents a statement or fact.

When processing RDF data, meta information, such as ontological structures and exact range definitions of predicates, are desirable and ideally provided by a knowledge base. However in the context of LOD, knowledge bases are often incomplete or simply not available. Even when a knowledge base is available, we often observe triples that violate its axioms. This inconsistency and lack of metadata impedes the utilization of LOD. Thus, it is useful to automatically generate meta information, such as ontological dependencies, range definitions, and topical associations of resources. Association rule mining is a promising approach to create such metadata, as we show in this article.

As resources can be connected through multiple predicates, co-occurring in multiple relations, frequencies and co-occurrences of statement elements become an interesting object of investigation for pattern analysis methods, such as association rule mining. Association rule mining has been widely studied in the context of basket analysis and sale recommendations [5]. In fact, association rules can be discovered in any domain, with many items or events among which interesting relationships can be inferred from co-occurrence of those items or events in existing subsets (transactions). To mine RDF data, several questions must be answered: What should be mined in which context, and what are the application fields for each approach. Previous work concentrates on inductive logic programming and graph mining, or is restricted to scenarios where domain knowledge and complete ontology structures are available.

We propose an approach that applies association rule mining at RDF-statement level by introducing the concept of *mining configurations*, briefly introduced in [2]. A min-

Z. Abedjan (✉) · F. Naumann
Hasso Plattner Institute, Potsdam, Germany
e-mail: ziawasch.abedjan@hpi.uni-potsdam.de

F. Naumann
e-mail: felix.naumann@hpi.uni-potsdam.de

ing configuration specifies one element of the SPO construct as the context of rule mining (the transaction identifiers) and another as the target of rule mining (the items and transactions). For each of the possible six configuration we describe the corresponding opportunities and application fields. In particular we show four applications that are based on mining configurations and benefit the usability and machine-readability of RDF data:

Predicate Suggestion When creating new triples manually, the creator might not exactly know which properties and values should be chosen. For instance, authors of Wikipedia infoboxes, which are the information source for DBpedia facts, are often inexperienced and only infrequently edit such data. Such users might forget to use certain predicates or might use similar but not common predicates for a new entry (e.g., city instead of locationCity). Those heterogeneous entries make integration of the complete dataset difficult. Predicate suggestion remedies the problem, providing users with a list of commonly used predicates, based on the already existing predicates for the entry.

Enrichment with Missing Facts Applying the suggestion approach to objects in addition to predicates enables us also to suggest object values as part of a new fact. We can then combine both approaches to *amend* data with completely new facts. As a proof-of-concept we applied our mining concept to the popular DBpedia data set [7] and achieved promising results.

Data-driven Ontology Re-engineering While there are best practices for publishing Linked Open Data using established ontologies [14], our analysis of the Billion Triple Challenge Dataset 2011 showed that due to various reasons certain "misusage" patterns occur frequently. This misuse partly stems from the fact that ontology definitions may be either too specific or too generic. The mismatch of data and ontology impedes the integration of data sources. We have identified two general misusage cases that can be remedied following our rule-based methodology [1].

Query Relaxation Through Predicate Expansion Another possibility to deal with ontology misusage in RDF data is to relax query results. Analyzing a SPARQL query workload provided by usewod2012[1] we encountered multiple sets of SPARQL queries that included UNION constructions joining dozens of patterns to account for schema and value errors and abbreviations. For example, a query for company entities labeled with IBM looked not only for the pattern

?company **dbpedia-prop:name** "IBM"@en

but also for

?company **rdfs:label** "IBM"@en,

via a UNION construction. We show a rule-based approach for discovering such pairs of predicates that can be used for query relaxation.

The remaining of this article is organized as follows: First we give an introduction to our methodology that was introduced in [2]. In Sect. 3 we show how rule mining can be applied for predicate suggestion and in Sect. 4 we show how the approaches can be extended for auto-amendment of new triples. In Sect. 5 we present our approach for dealing with inconsistent property usage through ontology re-engineering from [1] and in Sect. 6 how to deal with this problem through predicate expansion [3]. Then we give an overview of related work in Sect. 7 and conclude in Sect. 8.

2 Preliminaries

Our approach is based on association rule mining that is enabled by our concept of mining configurations [2]. First, we give a brief introduction to the concept of association rule mining. Next, we introduce our approach of mining configurations for RDF data and outline the characteristics of each configuration.

2.1 Association Rule Mining

The concept of association rules has been widely studied in the context of market basket analysis [4], however the formal definition is not restricted to any domain: Given a set of items $I = \{i_1, i_2, \ldots, i_m\}$, an association rule is an implication $X \rightarrow Y$ consisting of the *itemsets* $X, Y \subset I$ with $X \cap Y = \emptyset$. Given a set of transactions $T = \{t | t \subseteq I\}$, association rule mining aims at discovering rules holding two thresholds: minimum support and minimum confidence.

Support s of a rule $X \rightarrow Y$ denotes the percentage of transactions in T that include the union of the *antecedent* (left-hand-side itemset X) and *consequent* (right-hand-side itemset Y) of the rule, i.e., $s\%$ of the transactions in T contain $X \cup Y$. The confidence c of a rule denotes the statistical dependency of the *consequent* of a rule from the *antecedent*. The rule $X \rightarrow Y$ has confidence c if $c\%$ of the transactions T that contain X also contain Y. Algorithms to generate association rules decompose the problem into two separate steps:

1. Discover all frequent itemsets, i.e., itemsets that hold minimum support.
2. For each frequent itemset a generate rules of the form $l \rightarrow a - l$ with $l \subset a$ that hold minimum confidence.

[1] http://data.semanticweb.org/usewod/2012/.

Table 1 Six configurations of context and target

Conf.	Context	Target	Use case
1	Subject	Predicate	Schema discovery
2	Subject	Object	Basket analysis
3	Predicate	Subject	Clustering
4	Predicate	Object	Range discovery
5	Object	Subject	Topical clustering
6	Object	Predicate	Schema matching

While the second step of the algorithm is straightforward, the first step marks the bottleneck of any algorithm. The three best known approaches to this problem are Apriori [5], FP-Growth [13], and Eclat [23]. For each of these algorithms, there exist multiple modifications and optimizations. We use the FP-Growth algorithm for our intentions.

2.2 Mining Configurations

To apply association rule mining to RDF data, it is necessary to identify the respective item set I as well as the transaction base T and its transactions. Our mining approach is based on the subject-predicate-object (SPO) view of RDF data.

Any part of the SPO statement can be regarded as a *context*, which is used for grouping one of the two remaining parts of the statement as the *target* for mining. So, a transaction is a set of target elements associated with one context element that represents the transaction id (TID). We call each of those *context* and *target* combinations a *configuration*. Table 1 shows an overview of the six possible configurations and their preliminarily identified use-cases. Each can be further constrained to derive more refined configurations. For instance, the subjects may be restricted to be of type Person. In the following we further elaborate the meaning of each configuration with regard to the according target of mining (Table 1).

Mining Subjects In the RDF model, all statements with same subject represent one entity. Subjects with many common predicates can be considered as similar subjects. Thus, mining subjects in the context of predicates (Conf. 3) results in rules that express clustering or ontological affiliation of entities.

For instance, in the DBpedia set we retrieved rules between subjects that can be classified as presidents, musicians, or athletes, such as *George Washington* → *Lyndon B. Johnson* with 92 % confidence. As rules come up when subjects share a minimum number of properties, it can be expected that varying the support leads to clusterings and ontological concepts that differ in granularity. So lowering the minimum support leads to rules connecting athletes and presidents, because they share predicates of the more general concept "person", such as *name* or *birth_place*.

Mining subjects in the context of objects (Conf. 5), i.e., discovering subjects that share a minimum number of object values, results in rules between entities that are topically related. Objects are values that might be associated with subjects in different relations. E.g., several persons may share the object *Berlin* in different roles like *birth* or *death_place* or *home_town*. In fact, up to 50 distinct predicates in the DBpedia ontology infoboxes data set version 3.7 involve the city *Berlin* as object value. Therefore, organizations as well as persons and instances of other types might share the same objects, and are consequently topically related.

Mining Predicates While subjects represent entities in RDF data, predicates represent the schema for those entities. So, mining predicates in the context of subjects (Conf. 1) results in patterns and rules that show dependencies of schema elements among entities and can be used for schema discovery and analysis. Rules such as {*associatedBand, instrument*} → *associatedMusicalArtist* with 99 % confidence and 3 % support or {*activeYearsEndDate, party*} → {*activeYearsStart-Date, birthDate, successor*} with 2.5 % support and 68 % confidence show that the data contains different schemata for musicians and politicians. The difference in confidence might trigger a closer examination of possible reasons for loose or tight consistency of the schemata.

Mining predicates in the context of objects (Conf. 6) aims at discovering predicates that have a strong overlap in their value ranges. As predicates define the schema of entities, rules within this configuration can be used for schema matching or synonym discovery. For instance, we discovered rules between the predicates *associatedBand* and *associatedMusicalArtist* that have a confidence of 100 % in both directions.

Mining Objects In accordance to our view of entities and schemata, objects would represent the actual values that describe an entity. Thus, mining in the context of subjects (Conf. 2) means to discover patterns between values that are associated to each other by co-occurring for many entities. For example, the rule *Buenos Aires* → *Argentina* with 85 % confidence shows that entities associated with a capital town are probably also associated with the corresponding country. Our approach on enriching RDF data uses these strong rules.

Rules in the context of predicates (Conf. 4) imply range discovery of predicates as they connect values, such as numbers, countries, or cities. Exemplary rules include *1* → {*2, 3*} or *Albania* → *Italy*. In fact, the mining results of this configuration is very similar to the configuration for mining subjects in the context of predicates. Regarding the fact that subjects and objects have semantically different roles in a statement, it is worth reasoning about the actual difference of both configurations.

Table 2 Facts in SPO structure from DBpedia

Subject	Predicate	Object
Obama	birthPlace	Hawaii
Obama	party	Democrats
Obama	orderInOffice	President
Merkel	birthPlace	Hamburg
Merkel	orderInOffice	Chancellor
Merkel	party	CDU
Brahms	born	Hamburg
Brahms	type	Musician

Table 3 Configuration examples

TID	Transaction
(a) *Context: Subject, Target: Predicate*	
Obama	{*birthPlace, party, orderInOffice*}
Merkel	{*birthPlace, party, orderInOffice*}
Brahms	{*born, type*}
(b) *Context: Object, Target: Predicate*	
Musician	{*type*}
Hamburg	{*born, birthPlace*}
Hawaii	{*birthPlace*}
President	{*orderInOffice*}

In the following we exemplify the application of two configurations for mining predicates. Table 2 illustrates some SPO facts extracted from DBpedia. For legibility, we omit the complete URI representations of the resources and just give the human-readable values. The application of Configuration 1 from Table 1 to our example data set would transform the facts into three transactions, one for each distinct subject as illustrated in Table 3a. In this example, the itemset {*birthPlace, party, orderInOffice*} is a frequent itemset (support 66.7 %), implying rules such as *birthPlace* → *orderInOffice, party* and *orderInOffice* → *birthPlace, party* with 66.7 % and 100 % confidence, respectively. Furthermore, we can infere negative rules, such as *birthPlace* → ¬ *born*.

Configuration 6 in the context of objects would create the transactions presented in Table 3b. The frequent itemsets here contain predicates that are similar in their ranges, e.g., {*born, birthPlace*}. Given the negative rule in Conf. 1 and the pattern in Conf. 6, one could conclude that both predicates *born* and *birthPlace* have synonymous meanings and can be used for predicate expansion. In this work we present use cases based on the configurations 1, 2, and 6.

3 Predicate Suggestion

Suggestion of predicates or objects aims at two goals. First, the user who is creating new facts for a certain subject might be grateful for reasonable hints. Second, system feedback might prevent the user from using inappropriate synonyms for predicates as well as objects.

Suggestion Workflow For suggesting predicates or objects for a user that is creating facts for a certain subject we directly apply the Configurations 1 or 2, respectively. The major computation effort lays in generating all relevant association rules within the specific configuration. The suggestion workflow for predicates requires two preprocessing steps: (1) Generate all association rules between predicates. (2) Create an index to facilitate the retrieval of all relevant rules for a specific suggestion situation. We create a `rule matrix`, which is a two dimensional predicate-predicate matrix, where one index identifies the antecedents and one index the consequents of a rule. Each entry specifies the confidence of the rule involving the specific antecedent and consequent. For missing rules the entry is zero by default.

When a user is inserting or editing the facts related to a specific subject, the system is aware of all predicates that have already been inserted for the current subject. For generating a list of suggestions, all rules that incorporate the previously inserted predicates as their antecedents are retrieved. The suggestions then are all those predicates that occurred as consequences of the retrieved rules. The ranking of the suggestions is based on scores that are computed for each suggestion by aggregating all confidence values of the retrieved rules that have the specific predicate suggestion as their consequence. Based on the next chosen predicate the suggestion list changes again, because the rules that contain the new predicate as their antecedent are also taken into account.

Regarding our example from Table 2, imagine we are to insert a record for `D. Cameron` by beginning with the statements `"D. Cameron birthPlace London."` and `"D. Cameron orderInOffice Prime Minister."` All rules that have `birthPlace` and `orderInOffice` as their antecedents are retrieved. Considering only rules of size two, the set of predicate rules relevant for the next suggestion include `birthPlace` → `party` with 66.7 % confidence, `orderInOffice` → `party` with 100 % confidence, and `birthPlace` → `instrument` with 33.3 % confidence. Setting the `minConf` = 50 %, the predicate `party` would be the top suggestion.

Predicates vs. Objects The suggestion of objects is technically equivalent to the suggestion of predicates, but the amount of objects is by magnitudes larger. For instance, the DBpedia 3.6 data set contains 1,100 distinct predicates

Table 4 Evaluations for 10,000 predicate/object suggestions per data set

Type	p@5	p@10	MRR at 10
(a) Predicate suggestions			
Thing	0.420	0.639	0.20888
Person	0.510	0.714	0.26199
Place	0.507	0.771	0.21717
Work	0.275	0.555	0.12450
Species	0.648	0.909	0.27802
(b) Object suggestions			
Thing	0.050	0.055	0.03179
Person	0.027	0.028	0.02315
Place	0.069	0.069	0.06058
Work	0.015	0.015	0.01276
Species	0.440	0.541	0.22191

but 3,980,642 distinct objects. Thus, the minimum support threshold must be chosen very low to retrieve any rules from the data, but the frequencies of those values are too low to derive meaningful dependencies. Furthermore, for a user, choosing an object value for a proposed predicate is more convenient than vice versa. For example, a user might have created the entry `Barack Obama birthPlace Honolulu`. The system might contain an object-to-object rule with enough confidence saying `Honolulu` → `USA` and suggests to add a new fact with `USA` as its object. The user is confronted with the situation not knowing how the subject and the proposed object are connected and which predicate (`birthDate`, `residence`, etc.) to choose. In addition to the semantical fitting of the predicate, the user has also to consider its appropriateness with regard to consistency among similar entities, because ontological similar entities should share the same set of predicates.

Tables 4a and 4b show the performance of our algorithm on several datasets from DBpedia. Here we randomly removed predicates and objects from each entity and tried to suggest it with our algorithm. The tables display precision at x (p@x) and mean reciprocal rank (MRR) scores, showing that indeed association rule mining is much more suited for suggesting predicates than objects.

The question is now whether the suggestion of objects or predicates results in better results when we know not only the corresponding subject but also one of the remaining two parts. In the following section we show that indeed this intuition enables us to automatically generate completely new facts.

4 Auto-amendment of Triples

Following our suggestion scenario from Sect. 3, based on a subject that is being edited, the algorithm could try to generate new facts by guessing predicate and object combinations. We call this method of creating new statements where the user decides which subject has to be amended with new triples *user-driven auto-amendment*. The drawback of this approach is that no matter in which order the predicate and the object are being generated, the algorithm has difficulties to decide for the correct object, because there are only few high-confidence object rules. On the other hand experiments showed that knowing the correct object and subject of a fact, the precision for choosing the correct predicate with our approach is very high when applied to different samples of DBpedia 3.6 (see Table 5). Here, the suggestion algorithm considers only those predicates that have been witnessed with the given object in the data set. Based on this observed phenomenon we present a different way of creating new statements. Our *data-driven auto-amendment* approach lets the system itself choose the subjects that should be amended with new triples.

Table 5 Results for predicting 20,000 random predicates for each type

Type	Predictions	Precision	Recall
Thing	18,731	89.98 %	84.28 %
Place	19,775	92.84 %	91.80 %
Person	19,936	77.34 %	77.10 %
Work	19,865	87.04 %	86.55 %
Species	19,986	89.67 %	89.61 %
Organization	19,820	81.31 %	80.58 %
Animal	19,975	99.97 %	99.84 %
Album	19,861	96.27 %	95.61 %
Film	19,842	93.77 %	93.03 %

Our approach is based on the following intuitions:

1. For object rules $o_1 \rightarrow o_2$ with high confidence (above 90 %) the subjects occurring with the object o_1 are also likely to occur with the object o_2. However 10 % of the subjects that occur with o_1 *violate* the rule by not occurring with o_2 in any fact. Our assumption is that those facts are absent, because of missing thoroughness during data creation. For example a user that adds Honolulu as the `birthPlace` of a person assumes that the country where Honolulu lies (namely the USA) is implicitly given. Another example is the high-confidence rule `Wehrmacht` → `World War II`, which is comprehensible but raises the question why the confidence value is not 100 %, since per being registered as members of the Wehrmacht (the Nazi Germany army) should have accordingly taken part in World War II.
2. A subject should not be enriched with a fact containing object o_2 if on the basis of the rules involving schema predicates, no predicate can be chosen for the connection

with o_2. This intuition allows a softening of the earlier intuition that expects all subjects that violate $o_1 \rightarrow o_2$ should be extended with a triple containing o_2.

For data-driven auto-amendment we need to generate all predicate rules, corresponding to Conf. 1 from Table 1, and store them within a predicate-predicate rule matrix. Then we generate high-confidence object rules $o_i \rightarrow o_j$ in the manner of Conf. 2. For each object rule $o_i \rightarrow o_j$, all subjects that occur with the antecedent of a high-confidence rule but not with its consequent are retrieved. These subjects may be amended with new facts having the current object rule consequent o_j as their value. The choice for 90 % as the high confidence threshold is arbitrary. The higher this threshold is, the fewer new statements can be generated but higher precision is achieved. Then the algorithm proceeds with retrieving the candidate predicates that have o_j in their range. The score for each candidate predicate is then computed in the same manner as described for predicate suggestions based on given rules with schema predicates as antecedents.

Note the number of new facts depends on the number of existent high-confidence rules and their corresponding set of rule violating subjects. Using this approach on DBpedia v3.6 we were able to generate 26,646 new facts out of which 31 % where actually included in the later version 3.7. Most of the inclusions correspond to new facts on entities of types `Artist` or `Animal`, where the ratio was above 50 %.

The runtime of our algorithm even for the complete dataset is in the order of a few minutes (including the time for creating the transaction data base for association rule mining) and mostly depends on the parameters used for rule mining. The runtime reported here and in the rest of this article refer to experiments on a standalone notebook with a 2.66 GHz Intel Core Duo processor and 4 GB DDR3 memory.

5 Reconciling Ontologies and Data

A common case of inconsistency is the mismatch of ontology definitions and the underlying data. In particular, divergences between ontology specification and instance data may occur in two scenarios: On the one hand, the ontology might have been developed independently and before actual data using it was published, e.g., in the case of the "Friend of a Friend"[2] ontology (FOAF). On the other hand, the ontology might have been tailored for existing data, e.g., in the case of the DBpedia project, which evolved extensively since its first specification, while revising existing class definitions was sometimes neglected during this evolutionary process.

[2] http://xmlns.com/foaf/spec/.

Based on an existing ontology, we identify two typical cases where the specification differs from usage patterns: *overspecification* and *underspecification*. We refer to a certain class as being overspecified, if one or more properties are declared for this class by the ontology, but are rarely (if ever) used for real-world data, e.g., `scottishName` for `Settlement`. There are several reasons, why overspecification occurs: Data providers cannot set proper values for the defined properties, e.g., a `scottishName` for a non-Scottish `Settlement`. There may be multiple semantically equivalent properties defined for a class, e.g., `occupation` and `profession` for `Person`. Last but not least, subclasses might have been added and (inherited) properties are now used only in combination with these, e.g., `philosophicalSchool` used exclusively with `Philosopher`, but defined for parent class `Person`.

A class is underspecified, when in real-world data certain properties are used frequently even though they are not specified by the vocabulary. Underspecification may occur when the class definition lacks certain properties that are commonplace in instance data, e.g., `genre` for `Band`. It could also happen that a property is defined for certain subclasses whereas it covers additional instances of their common parent class, e.g., `numberOfStudents` is defined for `University`, but also used for instances of other subclasses of `EducationalInstitution` such as `College`. Note that a class can simultaneously be overspecified and underspecified (with regard to different properties).

Given a dataset with typed instances and a corresponding ontology, we apply frequency and association rule analysis by applying Conf. 1 to identify and remedy over- and underspecification [1]. Identifying cases of overspecification in a class definition is straightforward: Given a minimum support threshold of s, each property that does not hold s in the given data constitutes an overspecification of the current class and should therefore be marked as removal candidates for this class. Thus, for each property defined for the class its distinct occurrences as predicate for all entities of the given class are counted and the total is compared against the minimum support. We identified 503 removal suggestions in the DBpedia 3.6 ontology and 622 removal suggestions in the DBpedia 3.7 ontology, all with support $\leq$1 %. Table 6 shows sample results of overspecification in DBpedia 3.7. Some of the removal suggestions can be moved to a more suitable subclass as suggested in [1].

Having marked removal suggestions, we identify predicates that are used frequently for instances of a specific class but are not defined as properties for the class itself or any of its parent classes in the ontology. Based on association rules, we propose only those predicates to a class that are highly correlated with already (validly) defined properties of this class. This way we avoid suggesting frequent predicates that are used because of wrong type assignment of the associated

Table 6 Overspecified properties for DBpedia 3.7

Property	Class	Support
scottishName	Settlement	0.000 %
distanceToEdinburgh	Settlement	0.021 %
philosophicalSchool	Person	0.202 %
countySeat	PopulatedPlace	0.831 %
anthem	PopulatedPlace	0.147 %
depth	Place	0.723 %
numberOfGraduateStudents	EducationalInstitution	0.300 %

Table 7 Suggestion quality for DBpedia 3.6 and 3.7

DBpedia	Total	Useful	Not Useful	Undecided
3.6	283	234 (83 %)	15	34
3.7	317	268 (85 %)	31	18

instances, i.e., instances where no predicates is included in the class definition. Furthermore, the suggestions are more accurate within a branch of the subclass hierarchy.

The entire process described so far is in the order of a few hours (including the time for creating the transaction database for association rule mining) even for large datasets (such as the BTC crawl) and mostly depends on the parameters used for rule mining.

Table 7 illustrates the amount and quality of class property suggestions for DBpedia 3.6 and 3.7 (*minSupp*: 1 %, *minConf*: 70 %). Overall, the majority of the suggestions have been labeled as useful. Suggestions marked as undecided are those for which we could not decide whether they enhance the class definition or not. This was often the case, when a similar or synonymous property had already been defined for a class in the ontology (e.g., for `Person`, `Person/weight` is specified, `weight` is suggested). Synonym discrepancy constitutes a major problem for data consumers, as it happens that either properties are inconsistently used or the expectation of a user towards a property and the ontology designer may diverge. In the next section we pick up on the synonym discrepancy and present a solution for discovering such synonyms.

6 Predicate Expansion

We already showed that some discrepancies between property usage and ontology definitions emerge when instead of defined properties synonymous predicates are used in the data. Some examples that we encountered during our evaluations on the DBpedia data set where for instance `city` or `location` instead of `locationCity`. Of course two synonymous predicates may have been defined deliberately for two disjoint purposes, but because they have been used in substitution of each other, the data consumer has to deal with the inconsistency. We developed an approach for automatically discovering predicates that have been used in substitution of each other in the data, i.e., they have some synonymous meaning. The discovery of such dependencies is relevant for query expansion. A user that looks for actors of a movie and intuitively chooses the predicate `starring` will miss all actors where a synonymous predicate like `artist` has been used. Note, we explicitly talk about synonymously used predicates instead of synonym predicates. For example, predicates with more general or specific meaning often substitute each other in the data. E.g., `artist` is often used as a substitute for `starring` even though `artist` is more general than `starring`.

Our approach is again based on mining configurations. We apply Configurations 1 and 6 in the same manner as exemplified in Sect. 2. With Configuration 1 we perform schema analysis in the context of subjects. Configuration 6 enables us to mine similar predicates in the context of objects. We also looked into the range structure of predicates by looking at value type distributions. Despite the fact that type definitions might not always be available we could not identify any benefit to the range analysis approach in our experiments.

Configuration 1 enables us to do frequency analysis and rule discovery per entity. We used this configuration already for suggesting new predicates for data creators and generating inclusion suggestions for the ontology. Here we have to look at a different intuition: Expansion candidates for a predicate should not co-occur with it for any entity. It is more likely for entities to include only one representative of a synonymous predicate group within their schema, e.g., either `starring` or `artist`. That is why we look for negative correlations in Configuration 1.

Negative schema correlations might also lead to false positives, such as `recordLabel` and `author` as both occur for different entities. While songs have the predicate `recordLabel`, books have the predicate `author`. So a negative correlation is not a sufficient condition for a predicate to be expanded by another. Therefore we also take the range content of predicates into account.

Table 8 Discovered top 5 synonym pairs on DBpedia subsets

	DBpedia Work	DBpedia Organization
1.	artist, starring	city, location
2.	artist, musicComposer	city, hometown
3.	author, writer	location, hometown
4.	creator, writer	city, ground
5.	composer, musicComposer	city, locationCity

Our second intuition is that as synonym predicates have a similar meaning they also share a similar range of object values. Normally when trying to compute the value overlap between two predicates one would look at the ratio of overlaps depending on the total number of values of such a predicate. We apply a more efficient range content filtering approach (RCF) based on Conf. 6 that constitutes a mining scenario where each transaction is defined by a distinct object value. So each transaction consists of all predicates containing the distinct object value in their range. Frequent patterns in this configuration are sets of predicates that share a significant number of object values in their range. Experiments showed that this approach is by magnitudes faster than the pairwise overlap recognition approach.

For combining both configurations we decided on the following order: (1) first retrieve all predicate pairs through range content filtering, then (2) analyze their schema co-occurrences. This strategy has two advantages: as retrieving negative correlations and type vectors is time-consuming, it is reasonable to perform both on given candidates instead of using them on the complete data set to retrieve candidates. Furthermore, the minimum support threshold for range value overlap is a more expressive threshold than arbitrary correlation and scoring thresholds on schema level, which are more suited for ranking purposes of the filtered candidates.

We performed multiple experiments on DBpedia, GovWILD [8], and Magnatune.[3] Our experiments showed that our combined approach generates less false positives the more homogeneous the entities in the data set are. For example, on the Magnatune dataset that contained only music data we achieved precision values of 100 % on a 0.1 % support threshold for RCF, while on the DBpedia data set the precision was around 40 %. Here the algorithm generated false positives like `foundingPlace` and `birthPlace`, because the subjects of these predicates are from very different domains while the range of both very similar. The evidences by Configurations 1 and 6 are not enough. So it is vital to perform the algorithm on each domain of the data (Persons, Work, Places, Organizations) separately. Table 8 shows our top 5 results on the DBpedia Work and Organization data set. The total runtime of our algorithm including range content filtering and schema analysis is below 8 minutes for each presented dataset at a minimum support of 0.1 % for range content filtering and below 10 minutes at the threshold of 0.01 %. More details on the experiments can be found in [3].

[3] http://dbtune.org/magnatune/.

7 Related Work

We apply existing data mining algorithms to the new domain of LOD and proposed four different use cases on this basis. Therefore, we show an overview of related work with regard to data mining in the semantic web as well as the most related approaches to our presented use cases.

Mining the Semantic Web Most research on mining the semantic web is so far in the fields of inductive logic programming and approaches that make use of the description logic of a knowledge base [15]. Those approaches concentrate on mining answer-sets of queries towards a knowledge base. Based on a general reference concept, additional logical relations are considered for refining the entries in an answer-set. A statistical approach for mining the semantic web is proposed by Nebot et al. [19], where a SPARQL endpoint allows the user to define targets of mining in any desired graph context.

Looking at RDF data as graph where resources are connected via predicates as edges, another related field of research is mining frequent subgraphs or subtrees [16]. However, in LOD no two different nodes in an RDF graph have the same URI. For frequency analysis it is necessary to have a more abstract view on the resources. For example one could look only at the ontological types or data formats of resources.

Predicate Suggestion For a proof-of-concept use case we chose a predicate suggestion systems for RDF data. A similar use case has also been applied by the authors of [10] in the context of web-scale extraction of structured data for knowledge base creation. The authors introduce a system that helps database designers to create schemata by providing schema auto-completion based on extracted web tables from the WWW.

Auto-amendment of Triples In the fields of populating knowledge bases with missing facts, the iPopulator [17] and KYLIN [21] are related projects. Both concentrate on populating Wikipedia infoboxes using the Wikipedia text articles and existing infobox templates. Our approach does not use any external data source or structural information, such as ontologies or templates but only the existing RDF statements of the current RDF corpus.

Ontology Engineering The most related work in this field is the schema induction approach by Völker et al. [20]. The authors describe how association rules can be used to recreate axioms of the DBpedia ontology. Fleischhacker et al. extend this approach to discover also characteristics that are not predefined by an ontology, such as predicate symmetry, asymmetry, and disjointness [12]. Our work on improving ontologies differs as we create specific suggestions for changing the ontology of a data set by removing or adding properties. Several works in the field on ontology engineering aim at establishing and enriching ontology specifications by using machine learning techniques [9]. The authors of [18] present a semi-automatic approach for cross-domain ontology learning. Similarly, in [22] machine learning methods are employed to refine the definition of the Wikipedia infobox-class ontology.

Query Expansion and Synonym Discovery Research on query expansion includes stemming techniques, relevance feedback, and other dictionary based approaches [6]. On their technical level the approaches do not apply to our SPARQL scenario as we do not retrieve documents but structured entities. Elbassuoni et al. have already presented a query expansion approach based on language models [11]. Our approach is based on association rules and a more simplistic model and we were able to process large datasets, such as DBpedia, in a few minutes.

8 Conclusion

We examined association rule mining on statement level and introduced the concept of mining configurations. We applied our methodology to several use cases that detect and can prevent inconsistency in RDF data. We showed how one configuration can be used for predicate suggestion and ontology re-engineering. Further we introduced approaches for triple amendment and predicate expansion, based on combining two configurations. Association rule mining at RDF statement level is an interesting field for further research, as there are remaining configurations to be elaborated and combined for other interesting use cases. Beyond the various combination and refinement possibilities of configurations, the consideration of negative correlations and association rules might also harbor interesting insights and applications. Last but not least we will publish our mining framework for the sake of repeatability and to support the semantic web community.

References

1. Abedjan Z, Lorey J, Naumann F (2012) Reconciling ontologies and the web of data. In: Proceedings of the international conference on information and knowledge management (CIKM), New York, NY, USA, pp 1532–1536
2. Abedjan Z, Naumann F (2011) Context and target configurations for mining RDF data (2 pp.). In: Proceedings of the international workshop on search and mining entity-relationship data (SMER), Glasgow
3. Abedjan Z, Naumann F (2013) Synonym analysis for predicate expansion. In: Proceedings of the extended semantic web conference (ESWC), Montpellier, France
4. Agrawal R, Imieliński T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the ACM international conference on management of data (SIGMOD), Washington, DC, USA, pp 207–216
5. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the international conference on very large databases (VLDB), Santiago de Chile, Chile, pp 487–499
6. Baeza-Yates RA, Ribeiro-Neto B (1999) Modern information retrieval. Addison-Wesley/Longman, Boston
7. Bizer C, Lehmann J, Kobilarov G, Auer S, Becker C, Cyganiak R, Hellmann S (2009) DBpedia—a crystallization point for the web of data. J Web Semant 7:154–165
8. Böhm C, Freitag M, Heise A, Lehmann C, Mascher A, Naumann F, Ercegovac V, Hernandez M, Haase P, Schmidt M (2012) GovWILD: integrating open government data for transparency. In: Proceedings of the international world wide web conference (WWW). Demo
9. Buitelaar P, Cimiano P (eds) (2008) Ontology learning and population: bridging the gap between text and knowledge. Frontiers in artificial intelligence and applications, vol 167. IOS Press, Amsterdam
10. Cafarella MJ, Halevy A, Wang DZ, Wu E, Zhang Y (2008) WebTables: exploring the power of tables on the web. In: Proceedings of the VLDB endowment, vol 1, pp 538–549
11. Elbassuoni S, Ramanath M, Weikum G (2012) RDF Xpress: a flexible expressive RDF search engine. In: Proceedings of the 35th international ACM SIGIR conference on research and development in information retrieval. ACM, New York, p 1013
12. Fleischhacker D, Völker J, Stuckenschmidt H (2012) Mining RDF data for property axioms. In: Meersman R, Panetto H, Dillon T, Rinderle-Ma S, Dadam P, Zhou X, Pearson S, Ferscha A, Bergamaschi S, Cruz I (eds) On the move to meaningful internet systems: OTM 2012. Lecture notes in computer science, vol 7566. Springer, Berlin, pp 718–735
13. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceedings of the ACM international conference on management of data (SIGMOD), pp 1–12
14. Heath T, Bizer C (2011) Linked data: evolving the web into a global data space, 1st edn, Morgan & Claypool
15. Józefowska J, Lawrynowicz A, Lukaszewski T (2010) The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. Theory Pract Log Program 10:251–289
16. Kuramochi M, Karypis G (2001) Frequent subgraph discovery. In: Proceedings of the IEEE international conference on data mining (ICDM), Washington, DC, pp 313–320
17. Lange D, Böhm C, Naumann F (2010) Extracting structured information from Wikipedia articles to populate infoboxes. In: Proceedings of the international conference on information and knowledge management (CIKM). ACM, New York, pp 1661–1664
18. Maedche A, Staab S (2001) Ontology learning for the semantic web. IEEE Intell Syst 16:72–79
19. Nebot V, Berlanga R (2010) Mining association rules from semantic web data. In: Proceedings of the international conference on industrial engineering and other applications of applied intelligent systems (IEA/AIE), Cordoba, Spain, vol 2, pp 504–513

20. Völker J, Niepert M (2011) Statistical schema induction. In: Proceedings of the extended semantic web conference (ESWC), Heraklion, Greece, pp 124–138
21. Wu F, Weld DS (2007) Autonomously semantifying Wikipedia. In: Proceedings of the international conference on information and knowledge management (CIKM). ACM, New York, pp 41–50
22. Wu F, Weld DS (2008) Automatically refining the Wikipedia infobox ontology. In: Proceedings of the international world wide web conference (WWW), Beijing, China, pp 635–644
23. Zaki MJ (2000) Scalable algorithms for association mining. IEEE Trans Knowl Data Eng 12:372–390