Synonym Analysis for Predicate Expansion

Ziawasch Abedjan and Felix Naumann

Hasso Plattner Institute, Potsdam, Germany
{ziawasch.abedjan,felix.naumann}@hpi.uni-potsdam.de

Abstract. Despite unified data models, such as the Resource Description Framework (RDF) on structural level and the corresponding query language SPARQL, the integration and usage of Linked Open Data faces major heterogeneity challenges on the semantic level. Incorrect use of ontology concepts and class properties impede the goal of machine readability and knowledge discovery. For example, users searching for movies with a certain artist cannot rely on a single given property artist, because some movies may be connected to that artist by the predicate starring. In addition, the information need of a data consumer may not always be clear and her interpretation of given schemata may differ from the intentions of the ontology engineer or data publisher.

It is thus necessary to either support users during query formulation or to incorporate implicitly related facts through predicate expansion. To this end, we introduce a data-driven synonym discovery algorithm for predicate expansion. We applied our algorithm to various data sets as shown in a thorough evaluation of different strategies and rule-based techniques for this purpose.

1 Querying LOD

The increasing amount of Linked Open Data (LOD) is accompanied by an apparently unavoidable heterogeneity as data is published by different data publishers and extracted through different techniques and for different purposes. The heterogeneity leads to data inconsistencies and impedes applicability of LOD and raises new opportunities and challenges for the data mining community [16]. On the structural level, consistency already has been achieved, because (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/fact. This unified structure allows standard query languages, such as SPARQL, to be used. However for real applications also factual consistencies are relevant. 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 usually incomplete or simply not available. For example, our recent work showed that there is a mismatch between ontologies and their usage [1]. Evaluations on the DBpedia data set showed that some of the mismatches occurred, because predicates where used that were synonymous to a predicate defined by the ontology (e.g., city or location instead of locationCity). Of course two synonymous

P. Cimiano et al. (Eds.): ESWC 2013, LNCS 7882, pp. 140–154, 2013.

predicates may have been defined on purpose for two disjoint purposes, but because they have been used in substitution of each other, the data consumer has to deal with the inconsistency. As we analyzed a SPARQL query workload provided by usewod2012[1] we encountered multiple sets of SPARQL queries that included UNION constructions as illustrated in Table 1. These examples show that applications already try to deal with the predicate inconsistency within the data by expanding their queries with UNION constructions containing synonymously used predicates. These UNION constructions further join dozens of patterns intercepting schema and value errors and abbreviations.

In the fields of traditional information retrieval there are already intuitions and techniques for expanding keyword queries. They comprise techniques for synonym discovery, stemming of words, and spelling corrections. In this work we want to concentrate on the discovery of synonymously used predicates. The discovery of sameAs-links between subject/object resources has already been extensively subject of research. However the discovery of synonymously used predicates has not received any attention at all. 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.

Table 1. Joined patterns with UNION in DBpedia query logs

Pattern pairs containing synonymous predicates
?company **dbpedia-prop:name** "International Business Machines Corporation"@en ?company **rdfs:label** "International Business Machines Corporation"@en
?place **dbpedia-prop:name** "Dublin"@en. ?place **dbpedia-prop:officialName** "Dublin"@en.
?airport **onto:iataLocationIdentifier** "CGN"@en. ?airport **prop:iata** "CGN"@en.

Synonym discovery is further interesting for the general purpose of enriching an existing synonym thesaurus with new synonyms that have evolved through the time as multiple people use different terms for describing the same phenomenon. Because LOD is formatted in RDF, synonym candidate terms are easy to extract and easier to compare with regard to their contextual occurrence. Note, synonym discovery in unstructured data, such as web documents, needs to consider natural language processing rules. Last but not least, the discovery of synonym predicates benefits the usage of LOD. Furthermore, for many data sources meta-data is only poorly provided. Identifying synonymously used predicates can support the evaluation and improvement of the underlying ontology and schema definitions. Usage of global synonym databases is not sufficient and might lead to misleading facts in this scenario, because of the heterogeneity of LOD, as predicates are used in different knowledge bases for different purposes

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

by different data publishers. So a data-driven approach is necessary to dissolve the existing synonym dependencies.

In this paper, we present an approach for discovering predicate pairs that substitute each other in the data and are good candidates for query expansions. Our approach is based on aggregating positive and negative association rules at statement level based on the concept of mining configurations [2]. We only incorporate information based on the given RDF graph. As a proof-of-concept we applied our algorithm to several LOD sources including the popular DBpedia data set [8].

The rest of this paper is organized as follows: In the next section we present related work with regard to synonym discovery and schema matching. Next we present necessary foundations with regard to RDF and association rules. In Section 4 we describe our algorithm. We evaluate our approach and strategies in Section 5 and conclude in Section 6.

2 Related Work

We apply existing data mining algorithms to the new domain of LOD and propose predicate expansion approach based on synonym discovery. Therefore, we present related work with regard to data mining in the Semantic Web as well as existing applications in the fields of synonym discovery. As most of our techniques for synonym discovery derive from schema matching approaches, we also give an overview of relevant schema matching approaches.

2.1 Mining the Semantic Web

There is already much work on mining the Semantic Web in the fields of inductive logic programming and approaches that make use of the description logic of a knowledge base [21]. 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. This approach depends on a clean ontological knowledge base, which is usually not available. Furthermore, that approach ignores the interesting opportunities of mining of rules among predicates.

As RDF data spans a graph of resources connected by predicates as edges, another related field of research is mining frequent subgraphs or subtrees [17]. However, in LOD no two different nodes in an RDF graph have the same URI. Therefore, frequency analysis cannot be performed unless we assume duplicate entries in the data set. But if we consider the corresponding type of each URI pattern analysis can be performed, because multiple URIs belong to the same type. Thus, any graph mining would be restricted to type mining and not data mining.

Among profiling tools, ProLOD is a tool for profiling LOD, which includes association rule mining on predicates for the purpose of schema analysis [9]. An approach based on predicate mining was introduced for reconciling ontologies[1].

As similar approach was also used for schema induction [22].The method of mining association rules on predicates is also applied in our work, however we go further than just analyzing the schema and show a concrete application that is based on this method and show how it can be combined to rule mining scenarios that also involve the objects of RDF statements.

2.2 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. So far, Shady et al. have already presented a query expansion approach based on language models [12]. Our approach is based on association rules and a more simplistic model and we were able to process large datasets, such as DBpedia, in couple of minutes. Most existing work for discovering synonyms is based on different language processing and information retrieval techniques. A common approach is to look for co-occurrence of synonym candidates in web documents [7,23]. The idea behind this approach is that synonymous word co-occur in documents [15]. So they calculate the ratio of real co-occurrence of two terms and the independent occurrence of each term. Note that for these approaches there are already known candidate pairs that have to be validated. In our scenario this assumption does not hold, as we also have to retrieve the candidate pairs.

While Baronis work [7] concentrates on globally valid synonyms the authors of [23] address context sensitive synonym discovery by looking at co-clicked query results. Whenever the distance between two clusters of clicked query results is below a certain threshold, the query terms can be seen as synonyms.

The approaches so far are very different from our domain where we want to discover synonym schema elements in RDF data. An approach that has a similar characteristic is the synonym discovery approach based on extracted webtables [10]. The authors introduce a metric that enables to discover synonyms among table attributes. However their approach is quite restrictive: they assume a context attribute given for making attributes comparable. Furthermore, they ignore instance-based techniques as they process only extracted table schemata.

2.3 Schema Matching

Schema matching differs from synonym discovery within schemata in the sense that two schema elements may be synonyms but still may not share a remarkable number of values. On the other hands two attributes may share a lot of values but their corresponding labels may not be synonyms from a global point of view. Still approaches for the discovery of attribute matches and synonyms follow similar intuitions. According to the classification of Rahm and Bernstein [20], we would classify our approach as a mixture of an instance-based and a schema level matching algorithm. At schema level we apply existing techniques to RDF data and evaluate their effectivity.

Existing instance-based approaches are different from our work as they compare the content of each attribute column-wise [11,13,19]. Choosing features for matching is cumbersome and algorithms that look for value overlaps lack efficiency. We propose an association rule based approach that discovers overlaps between attribute values in an Rdf corpus.

One could also perform schema matching on element level by using dictionaries, however the performance of those approaches has been poor in real data scenarios [18]. In this paper we want to focus on mining based features for synonym discovery.

3 Preliminaries

Our approach is based on association rule mining that is enabled by two mining configurations introduced by [2]. First of all we give a brief introduction to concept of association rule mining. Next we introduce of mining configurations for Rdf data and outline how we apply them to our use case.

3.1 Association Rule Mining

The concept of association rules has been widely studied in the context of market basket analysis [3], however the formal definition is not restricted by 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 fraction 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 minimal support.
2. For each frequent itemset a generate rules of the form $l \rightarrow a - l$ with $l \subset a$, and check the confidence of the rule.

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 [4], FP-Growth [14], and Eclat [24]. For each of these algorithms, there exist multiple modifications and optimizations. We use the FP-Growth algorithm for our paper.

3.2 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.

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. 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

Our mining approach is based on the subject-predicate-object (SPO) view of RDF data as briefly introduced in [2]. 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.

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 3 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 constrained to be of type Person, as happens to be the case in our example.

The application of Configuration 1 from Tab. 3 to our example data set would transform the facts into three transactions, one for each distinct subject as illustrated in Tab. 4a. 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 Tab. 4b. 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.

Table 4. Configuration examples

(a) Context: Subject, Target: Predicate

TID	transaction
Obama	{*birthPlace, party, orderInOffice*}
Merkel	{*birthPlace, party, orderInOffice*}
Lennon	{*birthPlace, instrument*}

(b) Context: Object, Target: Predicate

TID	transaction
Musician	{*type*}
Hamburg	{*born, birthPlace*}
Hawaii	{*birthPlace*}
President	{*orderInOffice*}

4 Generation of Candidates for Predicate Expansion

Our approach aims at discovering all possible predicate pairs where each predicate could be the expansion of the other one. Having identified all such candidate pairs the expansion candidates of a given predicate can easily be retrieved by retrieving all pairs in which the respective to be expanded predicate occurs.

We introduce three basic strategies that we combine for the discovery of these candidate pairs. The first two strategies make direct usage of the mining configurations from Tab. 3. 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. Additionally, we look into range structure of predicates by looking at value type distributions. All three approaches are derived from existing schema-level and instance-based schema matching techniques.

4.1 Schema Analysis

Configuration 1 enables us to do frequency analysis and rule discovery per entity. For instance, positive rules between predicates can be used for re-validating existing ontologies [1]. In our use case we have 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. For this purpose we developed an FP-Growth [14] implementation that retrieves all negative correlations for a set of candidate pairs. The approach can also be used stand-alone looking at all possible pairs that have a negative correlation in the data set. Negative correlation can be expressed by several score functions. One could look at the bidirectional correlation coefficient or consider some kind of aggregations of the negative rules' confidence values. In the following we describe each of the used scoring functions at schema level.

Confidence Aggregations. The confidence *conf* of the rule $p_1 \rightarrow \neg p_2$ describes the probability c% of predicate p_2 not to occur for the same entity where p_1 occurs. We refer to these rules as negative rules. If p_2 was a rare predicate that, however, occurs always with p_1, $conf(p_1 \rightarrow \neg p_2)$ might be considerably high however $conf(p_2 \rightarrow \neg p_1)$ would be close to 0%. Therefore we need to aggregate both confidence values. We experimented using the three aggregations maximum, minimum, and F-Measure (harmonic mean).

Reversed Correlation Coefficient. The drawback of confidence aggregation is that the scoring ignores the overall relevance of a pair within a data set. We apply the formula given in [5], which measures the linear relationship between two predicates:

$$cCoeff(X,Y) = \frac{N \cdot supp(X,Y) - supp(X) \cdot supp(Y)}{\sqrt{supp(Y) \cdot (N - supp(Y)) \cdot supp(X) \cdot (N - supp(X))}}$$

where N denotes the total number of baskets in the mining configuration, which, for Configuration 1, is equivalent to the total number of entities (distinct subjects) in the data set. For ranking purposes we reverse the sign of $cCoeff(X, Y)$, as we want to have positive scores on negative correlations. We label the score reversed correlation coefficient (RCC).

***Syn*-Function.** In [10] the authors introduce the *syn*-function for synonym discovery in different webtables. Their assumptions are also that synonyms never occur together. In case of LOD 'never' is a too strong assumption. Furthermore, their score is based on a set of context attributes. Unfortunately the authors did not mention how to choose this context attribute, if domain knowledge is not given. Nevertheless, the intuition behind their function that two synonymous predicates have the same odds in occurring with other predicates can also be applied in our scenario. Thus, we also adapted this score function and compared the results to the scoring functions named before.

Bare schema analysis leads to results also including incorrect pairs, 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. The context or the range of the predicates should also be taken into account. In the following we describe our strategies that complement the schema analysis by considering also the range of predicates.

4.2 Range Content Filtering

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 mining configurations (see Sec. 3.2).

Configuration 6 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 there range. Frequent patterns in this configuration are sets of predicates that share a significant number of object values in their range. As each configuration is an adaption of frequent itemset mining the threshold that decides whether two predicates are similar or not is minimum support and depends on the number of all baskets or all existing distinct objects. Furthermore, our approach ignores value overlaps that occur due to multiple occurrence of one distinct value in the ranges. We analyze the effect of these differences and show that our approach is much more efficient without any loss of quality. Similar to the schema analysis strategy also the range content filtering based on value overlaps is not a sufficient condition for discovering synonymously used predicates. For example the predicates `birthPlace` and `deathPlace` share a remarkable percentage of their ranges but are obviously not

used synonymously. However this candidate pair can be pruned looking at their exclusion rate per entity during schema analysis.

4.3 Range Structure Filtering

In some scenarios value range content filtering might not be the most appropriate technique as it requires two synonym predicates to share a portion of exactly equal values. However, real world data might contain synonymous predicates with completely disjoint range sets where the range elements are only ontologically similar. This is often the case when looking at predicates describing numbers and dates. Therefore existing work looks not only at exact overlaps but also on general string or token characteristics, such as string length and character distributions [11,19]. As the goal of this paper is to analyse the capabilities graph data and statement level mining, we do not dive into literal similarities and character distributions. Furthermore our experiments showed that based on range similarity we are already able to discover all pairs that contain values with similar ranges. Instead we look at type distributions in predicate ranges. So for every object in the range of a predicate we retrieve its type from the graph. Then we create type vectors per predicate where each component contains the number of the occurrences of one type. As each entity in RDF might have several types due to existing type hierarchies, i.e., Barack Obama is a Politician as well as a Person, we considered only the most specific type of an entity.

Having type vectors for a predicate pair, the range type similarity can be computed using measures, such as cosine similarity or weighted Jaccard similarity. Preliminary experiments showed that weighted Jaccard similarity seems more promising because cosine similarity results into high scores as soon as one component value of one vector is very large although all other components have very small values. Missing type values, e.g., in case of dates and other numerical values, have been handled as unknown types, whereas no two unknown types are equal.

4.4 Combined Approach

We have introduced three different ways of generating and evaluating synonym candidate pairs. It is crucial to find a reasonable order for combining those three to make best use of the intuitions and achieve optimal quality and to be efficient at the same time. We decided on the following order: (1) first retrieve all predicate pairs through range content filtering, (2) filter those pairs by range structure filtering and then (3) 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 overlapping 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. Consider that type range filtering can be applied only to data sets for which the type information is available. In our experiments

we could use the type filtering approach only for the DBpedia data, and even there it did not contribute to the precision on top of range content filtering.

5 Evaluation

We evaluated our approach with regard to precision and recall of generated expansion candidates. Table 5 shows the data sets with the corresponding numbers of distinct triples, subjects, predicates, and objects used for experiments. Because DBpedia contains data from different domains, we also performed our experiments on subsets of a certain domain, such as people and places. In the following we first show to which extent each component of our algorithm contributes to the quality of query expansion candidate analysis. Then we show overall precision results on multiple data sets. Last, we illustrate the efficiency gain of our frequent itemset based overlap discovery method towards the standard value-overlap approach.

Table 5. Datasets for evaluations

Source	#triples	#predicates	#subjects	#objects
Magnatune	243,855	24	33,643	68,440
Govwild	7,233,610	35	963,070	2,648,360
DBpedia 3.7	17,518,364	1,827,474	1,296	4,595,303
DBpedia Person	4,040,932	237	408,817	982,218
DBpedia Organisation	1,857,849	304	169,162	731,136
DBpedia Work	2,611,172	136	262,575	751,916

5.1 Step-Wise Evaluation of Recall and Precision

To evaluate the components of our algorithm, it is necessary to be able to classify good and poor expansion candidates. For this purpose, we manually classified 9,456 predicates pairs of a dataset. The classification of predicate pairs for expansion appropriateness is cumbersome, because one has to look for defined ranges, example values, and consider query intentions using these predicates. We chose the data sets with the lowest number of predicates, Magnatune, and the data set comprising all entities of type

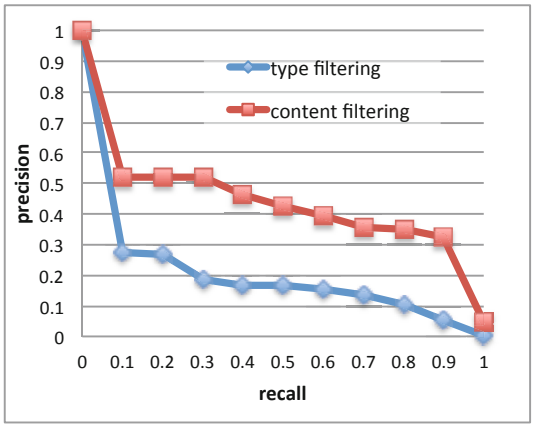
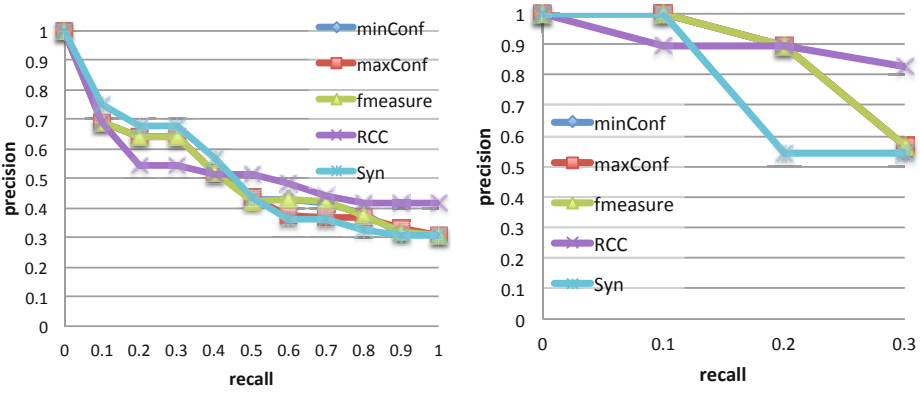


Fig. 1. Precision recall curve for the filtering methods

Work from DBpedia. A predicate pair is annotated as a correct expansion pair if both predicates are appropriate candidates for expanding the respective other predicate. Each classification result was validated by three experts (computer scientists). All in all, we discovered 82 expansion candidate pairs among the predicates for Work entities and 9 candidates in the Magnatune data set, out of 9,180 and 276 pairs, respectively.

First, we evaluated the precision/recall curve of the range-content and the range-type filtering approaches on the Work dataset as illustrated in Fig. 1. For this purpose we sorted all pairs twice, once with regard to their support value and once with regard to the weighted Jaccard distance of their range types. As the diagram illustrates, both approaches perform better than a random approach, which results in 0.8% precision on a recall of 100%. However, the precision of the range-content filtering method is on all recall levels better than the precision achieved with range-type filtering.



(a) 0.01% support (b) 0.1% support

Fig. 2. Precision recall curve of schema scores on the Work dataset

Figures 2a and 2b illustrate the ranking improvement of the algorithm using the schema scores. We chose the support thresholds 0.1% and 0.01% where the content filtering part resulted in 52% and 22% precision and recall levels of 28% and 98% respectively (see Fig. 1). At the support threshold of 0.01% the range content filtering achieved 22% precision and 98% recall. Figure 2a shows that all schema scores result in better precision on this recall level. Furthermore, on lower recall levels the precision is higher by orders of magnitudes. The precision improvement can be explained through the fact that predicate pairs with a very similar range but different semantics, such as album and previousWork, achieve lower scores on schema level as they often appear together. Looking at Fig. 2b the only difference is that at the highest possible recall level of 28% only the RCC score leads to better results. In general it can be observed that at high recall levels the RCC score performs better than all other scoring functions, while on low recall levels the *Syn* function performs better.

Regarding the results for Magnatune in Fig. 3, we can observe very high precision values even with range content filtering. However, even at a support threshold of 0.0% the schema-based scoring functions all perform better. If we raise the minimum support threshold to 0.01% or 0.1%, the precision remains 100% for all approaches, however the recall falls to 89% and 44%, respectively.

Next, we evaluate the precision of our combined approach on these two minimum support thresholds and fixed schema scoring thresholds.

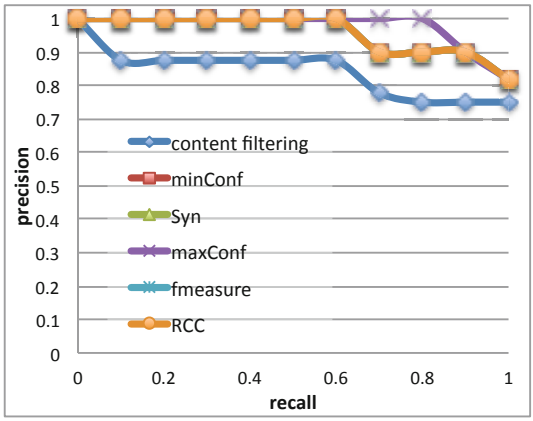


Fig. 3. Precision recall on Magnatune with 0.0% minimum support

5.2 Precision Quality

To evaluate the different approaches we defined minimum thresholds as follows: For the minimum, maximum, and f-measure confidence scores we fixed the threshold at 50% minimum confidence. For the RCC and *Syn* scores we set the threshold as >0.0. For RCC only scores above 0.0 indicate any negative correlation. The closer the value is to 0.0 the more random is the co-occurrence of two predicates. The *Syn* function results in scores above zero only if there is a significant correlation of the predicates. However, because the value is not normalized within a certain range, there is no basis for the choice of a higher threshold. That is why we use here the absolute value 0.0 as a threshold.

Comparing both Tables 6 and 7 one can see the precision improvement by leveraging the support threshold for RCF. Furthermore, one can observe that all schema scores behave very similar. The only significant differences can be observed for the Govwild data set, where minimum and f-measure confidence retrieve no correct results at all. The reason is that the Govwild dataset comprises data from different domains, such as people, locations, and organisations. That leads to false positives like name and city, because both people and organisations are connected to a city with the city attribute, while triples with cities as their subject use name for labeling the same city RDF object. The same reason also applies to the experiments on the complete DBpedia 3.7 data set. Looking at more specific domain data, such as Magnatune or DBpedia Work and Organisation the results are much better. Of course the numbers of retrieved results are much smaller, because the algorithm was able to filter nearly all true negatives.

One can conclude that the more cautious the thresholds are chosen the better quality can be achieved on all data sets. On data sets containing entities of very different domains, the algorithm produces too many false positives, so it is

Table 6. Precision at 0.01% RCF minimum support

Dataset	minConf	maxConf	f-Measure	RCC	Syn	RCF	# RCF results
Magnatune	**100%**	87.5%	**100%**	**100%**	87.5%	87.5%	8
Govwild	0%	**20%**	0%	14%	0%	20%	25
DBpedia 3.7	**32%**	**32%**	**32%**	15%	22%	32%	1115
DBpedia Person	32%	32%	32%	**35%**	26%	32%	308
DBpedia Work	49%	52%	50%	**61%**	60%	22%	256
DBpedia Organisation	**33%**	32%	32%	31%	32%	32%	412

Table 7. Precision values at 0.1% range content filtering minimum support

Dataset	minConf	maxConf	fMeasure	RCC	Syn	RCF	# RCF results
Magnatune	**100%**	**100%**	**100%**	**100%**	**100%**	100%	4
Govwild	0%	**56%**	0%	50%	0%	50%	10
DBpedia 3.7	40%	43%	38%	**46%**	45%	36%	64
DBpedia Person	56%	49%	50%	**60%**	-	40%	35
DBpedia Work	73%	57%	74%	78%	**89%**	52%	46
DBpedia Organisation	88%	86%	90%	89%	**95%**	85%	45

Table 8. Runtime experiment results

Dataset	RCF @ 0.1% support	RCF @ 0.01% support	naive RCF
Magnatune	4,116 ms	4,417 ms	18,122 ms
Govwild	66,297 ms	67,676 ms	> 3h
DBpedia Work	93,876 ms	97,676 ms	> 3h
DBpedia 3.7 (complete)	122,412 ms	127,964 ms	> 3h

always reasonable to perform the algorithm on each domain fraction of the data set separately. Performing the experiments on entities of the more specific type Actor that is a subclass of Person, we achieved much better precision, e.g., RCF and RCC values were 65% and 87% respectively.

5.3 Efficiency Analysis

We stated that our RCF approach for discovering value overlaps using Configuration 6 (see Sec. 3.2) is more efficient than pairwise comparison of predicates. Table 8 illustrates some runtime comparisons; we aborted runs after three hours. Our mining-based RCF approaches are always by faster than the naïve overlap approach by orders of magnitude, because predicate pairs with no overlap are filtered early. Furthermore the runtime of our approach is adaptive to support thresholds in the manner of frequent item mining, as it filters predicate pairs below the specified threshold in beforehand.

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%. The experiments have been performed on a notebook with a 2.66 GHz Intel Core Duo processor and 4 GB DDR3 memory.

6 Conclusion

In this paper we addressed data inconsistencies due to synonymously used predicates in RDF and introduced the concept of predicate expansion for SPARQL patterns. We presented several strategies for automatically discovering expansion candidates. We showed the strength and weakness of the strategies on different datasets, proposing a stacked algorithm based on range content filtering and schema analysis. Our evaluation showed that our algorithm performs very good on data containing only subjects of one domain, but produces more false positives on RDF data where the subjects represent entities of many different types. We believe that providing an optional predicate expansion interface at SPARQL endpoints is useful. An alternative approach is to (semi-)automatically remove or change facts with wrongly used predicates, based on the results of our synonym discovery.

References

1. Abedjan, Z., Lorey, J., Naumann, F.: Reconciling ontologies and the web of data. In: Proceedings of the International Conference on Information and Knowledge Management (CIKM), Maui, Hawaii, pp. 1532–1536 (2012)
2. Abedjan, Z., Naumann, F.: Context and target configurations for mining RDF data. In: Proceedings of the International Workshop on Search and Mining Entity-Relationship Data (SMER), Glasgow (2011)
3. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), Washington, D.C., USA, pp. 207–216 (1993)
4. Agrawal, R., Srikant, R.: 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 (1994)
5. Antonie, M.-L., Zaïane, O.R.: Mining positive and negative association rules: An approach for confined rules. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 27–38. Springer, Heidelberg (2004)
6. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)
7. Baroni, M., Bisi, S.: Using cooccurrence statistics and the web to discover synonyms in technical language. In: International Conference on Language Resources and Evaluation, pp. 1725–1728 (2004)
8. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - A crystallization point for the Web of Data. Journal of Web Semantics (JWS) 7, 154–165 (2009)

9. Böhm, C., Naumann, F., Abedjan, Z., Fenz, D., Grütze, T., Hefenbrock, D., Pohl, M., Sonnabend, D.: Profiling linked open data with ProLOD. In: Proceedings of the International Workshop on New Trends in Information Integration (NTII), pp. 175–178 (2010)
10. Cafarella, M.J., Halevy, A., Wang, D.Z., Wu, E., Zhang, Y.: WebTables: exploring the power of tables on the web. Proceedings of the VLDB Endowment 1, 538–549 (2008)
11. Doan, A., Domingos, P., Halevy, A.Y.: Reconciling schemas of disparate data sources: a machine-learning approach. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), New York, NY, pp. 509–520 (2001)
12. Elbassuoni, S., Ramanath, M., Weikum, G.: Query relaxation for entity-relationship search. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 62–76. Springer, Heidelberg (2011)
13. Gottlob, G., Senellart, P.: Schema mapping discovery from data instances. Journal of the ACM 57(2), 6:1–6:37 (2010)
14. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), pp. 1–12 (2000)
15. Harris, Z.: Distributional structure. Word 10(23), 146–162 (1954)
16. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Morgan Claypool Publishers (2011)
17. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), Washington, D.C., pp. 313–320 (2001)
18. Li, W.-S., Clifton, C.: Semint: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. Data and Knowledge Engineering (DKE) 33(1), 49–84 (2000)
19. Naumann, F., Ho, C.-T., Tian, X., Haas, L.M., Megiddo, N.: Attribute classification using feature analysis. In: Proceedings of the International Conference on Data Engineering (ICDE), p. 271 (2002)
20. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal 10(4), 334–350 (2001)
21. Rettinger, A., Lösch, U., Tresp, V., d'Amato, C., Fanizzi, N.: Mining the semantic web - statistical learning for next generation knowledge bases. Data Min. Knowl. Discov. 24(3), 613–662 (2012)
22. Völker, J., Niepert, M.: Statistical schema induction. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 124–138. Springer, Heidelberg (2011)
23. Wei, X., Peng, F., Tseng, H., Lu, Y., Dumoulin, B.: Context sensitive synonym discovery for web search queries. In: Proceedings of the International Conference on Information and Knowledge Management (CIKM), New York, NY, USA, pp. 1585–1588 (2009)
24. Zaki, M.J.: Scalable Algorithms for Association Mining. IEEE Transactions on Knowledge and Data Engineering (TKDE) 12, 372–390 (2000)