University of Potsdam
Hasso Plattner Institute
Information Systems Group

Improving RDF Data with Data Mining

Ziawasch Abedjan

Hasso-Platter-Institut Potsdam

Universität Potsdam

A thesis submitted for the degree of

Dr. rer. nat.

March 2014

Reviewers

Prof. Dr. Felix Naumann
Hasso-PLattner-Institut Potsdam

Prof. Dr. Heiner Stuckenschmidt
Institut für Informatik und Wirtschaftsinformatik, Universität Mannheim

Prof. Dr. York Sure-Vetter
GESIS - Leibnitz-Institut für Sozialwissenschaften, Mannheim

Published online at the
Institutional Repository of the University of Potsdam:
URL http://opus.kobv.de/ubp/volltexte/2014/7133/
URN urn:nbn:de:kobv:517-opus-71334
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-71334

Abstract

Linked Open Data (LOD) comprises very many and often large public data sets and knowledge bases. Those datasets are mostly presented in the RDF triple structure of subject, predicate, and object, where each triple represents a statement or fact. Unfortunately, the heterogeneity of available open data requires significant integration steps before it can be used in applications. Meta information, such as ontological definitions and exact range definitions of predicates, are desirable and ideally provided by an ontology. However in the context of LOD, ontologies are often incomplete or simply not available. Thus, it is useful to automatically generate meta information, such as ontological dependencies, range definitions, and topical classifications.

Association rule mining, which was originally applied for sales analysis on transactional databases, is a promising and novel technique to explore such data. We designed an adaptation of this technique for mining RDF data and introduce the concept of “mining configurations”, which allows 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. To this end, we present rule-based approaches for auto-completion, data enrichment, ontology improvement, and query relaxation. Auto-completion remedies the problem of inconsistent ontology usage, providing an editing user with a sorted list of commonly used predicates. A combination of different configurations step extends this approach to create completely new facts for a knowledge base. We present two approaches for fact generation, a user-based approach where a user selects the entity to be amended with new facts and a data-driven approach where an algorithm discovers entities that have to be amended with missing facts.

As knowledge bases constantly grow and evolve, another approach to improve the usage of RDF data is to improve existing ontologies. Here, we present an association rule based approach to reconcile ontology and data. Interlacing different mining configurations, we infer an algorithm to discover synonymously used predicates. Those predicates

can be used to expand query results and to support users during query formulation.

We provide a wide range of experiments on real world datasets for each use case. The experiments and evaluations show the added value of association rule mining for the integration and usability of RDF data and confirm the appropriateness of our mining configuration methodology.

Zusammenfassung

Linked Open Data (Lod) umfasst viele und oft sehr große öffentlichen Datensätze und Wissensbanken, die hauptsächlich in der RDF Triplestruktur bestehend aus Subjekt, Prädikat und Objekt vorkommen. Dabei repräsentiert jedes Triple einen Fakt. Unglücklicherweise erfordert die Heterogenität der verfügbaren öffentlichen Daten signifikante Integrationsschritte bevor die Daten in Anwendungen genutzt werden können. Meta-Daten wie ontologische Strukturen und Bereichsdefinitionen von Prädikaten sind zwar wünschenswert und idealerweise durch eine Wissensbank verfügbar. Jedoch sind Wissensbanken im Kontext von Lod oft unvollständig oder einfach nicht verfügbar. Deshalb ist es nützlich automatisch Meta-Informationen, wie ontologische Abhängigkeiten, Bereichs-und Domänendefinitionen und thematische Assoziationen von Ressourcen generieren zu können.

Eine neue und vielversprechende Technik um solche Daten zu untersuchen basiert auf das entdecken von Assoziationsregeln, welche ursprünglich für Verkaufsanalysen in transaktionalen Datenbanken angewendet wurde. Wir haben eine Adaptierung dieser Technik auf Rdf Daten entworfen und stellen das Konzept der Mining Konfigurationen vor, welches uns befähigt in Rdf Daten auf unterschiedlichen Weisen Muster zu erkennen. Verschiedene Konfigurationen erlauben uns Schema- und Wertbeziehungen zu erkennen, die für interessante Anwendungen genutzt werden können. In dem Sinne, stellen wir assoziationsbasierte Verfahren für eine Prädikatvorschlagsverfahren, Datenvervollständigung, Ontologieverbesserung und Anfrageerleichterung vor.

Das Vorschlagen von Prädikaten behandelt das Problem der inkonsistenten Verwendung von Ontologien, indem einem Benutzer, der einen neuen Fakt einem Rdf-Datensatz hinzufügen will, eine sortierte Liste von passenden Prädikaten vorgeschlagen wird. Eine Kombinierung von verschiedenen Konfigurationen erweitert dieses Verfahren sodass automatisch komplett neue Fakten für eine Wissensbank generiert werden. Hierbei stellen wir zwei Verfahren vor, einen nutzergesteuerten

Verfahren, bei dem ein Nutzer die Entität aussucht die erweitert werden soll und einen datengesteuerten Ansatz, bei dem ein Algorithmus selbst die Entitäten aussucht, die mit fehlenden Fakten erweitert werden.

Da Wissensbanken stetig wachsen und sich verändern, ist ein anderer Ansatz um die Verwendung von RDF Daten zu erleichtern die Verbesserung von Ontologien. Hierbei präsentieren wir ein Assoziationsregeln-basiertes Verfahren, der Daten und zugrundeliegende Ontologien zusammenführt. Durch die Verflechtung von unterschiedlichen Konfigurationen leiten wir einen neuen Algorithmus her, der gleichbedeutende Prädikate entdeckt. Diese Prädikate können benutzt werden um Ergebnisse einer Anfrage zu erweitern oder einen Nutzer während einer Anfrage zu unterstützen.

Für jeden unserer vorgestellten Anwendungen präsentieren wir eine große Auswahl an Experimenten auf Realweltdatensätzen. Die Experimente und Evaluierungen zeigen den Mehrwert von Assoziationsregeln-Generierung für die Integration und Nutzbarkeit von RDF Daten und bestätigen die Angemessenheit unserer konfigurationsbasierten Methodologie um solche Regeln herzuleiten.

Acknowledgements

This thesis would not have been the same without the influence and support of many wonderful people around me. First of all, I would like to give special thanks to my advisor, Prof. Dr. Felix Naumann. Despite his valuable subject-specific feedback, which I received whenever needed, he kindly advised me on planning my research and research carrier.

Secondly, I want to thank all my colleagues and mentors, who I worked with during the last three years. In particular, I want to express my gratitude to my colleagues from the Information Sytsems Group at HPI. Especially I would like to thank Toni Grütze, Arvid Heise, and Johannes Lorey, with whom I worked successfully in several projects. I will also remember Tobias Vogel, Dustin Lange, Anja Jentzsch, Gjergji Kasneci, and Ralf Krestel for their advice and fruitful exchange of ideas. I would like to thank Jorge-Arnulfo Quiané-Ruiz, my advisor during my stay at QCRI, for showing me additional ways of thinking on research and engineering. I would also like to thank the HPI Research School for funding my research and providing the opportunity to connect with mentors and colleagues from multiple disciplines in computer science.

Thirdly, I would like to thank my friends for their moral support. Special thanks to my friend Azadeh, who decisively supported and encouraged me during the final stage of this work.

Last but not least, I want to thank my parents for simply everything.

Contents

For my mother,
Farah

Chapter 1

The Web of Data

The rapid growth of the World Wide Web (WWW) provides access to all sorts of resources and applications for more and more users. While in the beginning simple web documents provided niches for hobby surfers, with the technological advances, professional companies and organizations realized that the WWW provides new opportunities in order to connect to a larger audience and in particular to more customers.

With the rise of weblogs, wikis, and social networks, a wider range of Internet users have been encouraged to generate new content, leading into the Web 2.0 era. One of the most notable achievements of this era is Wikipedia, the free and online encyclopedia, which contains user-generated articles in more than 280 languages. Via hypertext links and search engines, users are able to swiftly navigate to the desired information and applications.

Today, one of the major challenges is to apply the same principles that made the WWW popular to data and content [Bizer et al., 2009a]. This challenge contributed to one of the major rivalries of the recent years in web search and information retrieval: between classic document search and entity search. The advocates of entity search often aim at a new vision of the World Wide Web: its transformation to the Semantic Web. The Semantic Web, in a nutshell, is the vision of making semantics of data representable and machine readable. Therefore, the aim of the Semantic Web is to convert the current web data, represented by unstructured and semi-structured documents into a *web of data*[1]. For the vision of *web of data* to become reality, two requirements must be met:

- Data should be openly available in a standard format: *Open Data*
- Relationships among data entries and across datasets should be available, too: *Linked Data*

[1]`http://www.w3.org/2001/sw/`

Even though visions of the Semantic Web community about the *web of data* have been tackled mostly in the academic area, there are several commercial approaches that encourage advancement in the field of entity-relationship data. Recently, the Google knowledge graph has been launched. For a given search term that can be mapped to a known entity, such as famous people or places, the search engine not only retrieves the most relevant web documents, but also a set of structured information that generally describe the entity. The same concept was applied to Wikipedia's infoboxes, which contain the most relevant information about an entity in a property-value manner within a structured table, which is easily perceivable by a user.

The backbone of the Semantic Web is the Resource Description Framework (RDF), which enables its current motor, the Linked Open Data (LOD) movement, to progress on its path. To advance the LOD vision, the research community has to deal with many hard challenges. In the following, we first give a brief introduction to RDF knowledge bases and LOD. Then we outline existing challenges and solutions, and list our contributions, which are elaborated in this thesis.

1.1 RDF Knowledge Bases and Linked Open Data

The Resource Description Framework (RDF) is a standard model for representing data in the Semantic Web[1]. In RDF, information about a real-world entity is represented in a triple structure consisting of a subject, a predicate, and an object (SPO). Throughout this thesis, we refer to data that is represented in RDF simply as RDF data. Each triple represents a statement or fact about that entity, which occurs as the subject of the fact. Table 1.1 denotes a set of facts about Barack Obama. Datasets that adhere to the RDF structure are also referred to as RDF *knowledge bases*, because they contain facts about the real world in a format that allows reasoning and inference of new knowledge [Hayes-Roth et al., 1983].

Subject	**Predicate**	**Object**
Barack Obama	birthPlace	Hawaii
Barack Obama	party	Democrat
Barack Obama	orderInOffice	President of the USA

Table 1.1: Facts in SPO structure

Originally relevant to denote meta data and logical expressions, the RDF data model significantly gained importance through the development of LOD. When Tim Berners Lee proposed to publish structured data that can be linked and

[1]`http://www.w3.org/RDF/`

shared freely over the web, coining the term "Linked Open Data", the Resource Description Framework became a fundamental concept for representing entity-relationship data. Along with RDF, the *SPARQL Protocol and* RDF *Query language* (SPARQL) became a quasi standard technology for querying RDF data.

For an RDF knowledge base to become linked data, it is important that the resources and vocabulary are represented via URIs and therefore dereferencable. In particular the following set of rules have been defined that serve as best practices for publishing Linked Data[1]:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information using the standards (RDF, SPARQL)
4. Include links to other URIs, so that they can discover more things.

For example the first fact from Table 1.1, has the following representation in the DBpedia knowledge base:

```
<http://dbpedia.org/resource/Barack_Obama>
 <http://dbpedia.org/ontology/birthPlace>
  <http://dbpedia.org/resource/Hawaii>.
```

Looking at the URIs, one can see that the predicate URI differs in its path elements from the subject and object URIs. This distinction stems from the fact that in knowledge bases, assertions such as types/classes membership of resources, subclass hierarchies and class properties are defined by an ontology. Those ontologies are usually defined in the Web Ontology Language (OWL) or RDF Schema (RDFS). It is important to note that while we refer to properties when talking about entities and classes, we use the term predicate when referring to in RDF triples.

According to `www.linkeddata.org`, up to now more than 300 data sources have been published and connected via `sameAs`-links that connect same world entities across datasets. One of the most notable datasets is DBpedia [Bizer et al., 2009b]. DBpedia is an RDF structured dataset that contains extracted data from Wikipedia and its infoboxes. Because the DBpedia dataset covers many domains, many data publishers link their data to DBpedia. The advancement in linking the datasets helps to disambiguate entities and reconcile knowledge from different sources. Online news sites, such as BBC sports and New York Times, already use Linked Data to annotate their web pages and to disambiguate their web identifiers for every entity and in turn provide their own collected knowledge as RDF datasets to the community.

[1]`http://www.w3.org/DesignIssues/LinkedData.html`

1.2 Challenges of Open RDF Data

The main goal of the Semantic Web is to improve the machine-readability of data. The Linked Open Data principles already contribute to the machine-readability by declaring structural standards, such as RDF, for representing knowledge. Especially, by defining that resources should be represented via HTTP URIs, data sources can be linked, which in turn eases the reconciliation of related information about real-world entities across multiple datasets [Heath and Bizer, 2011].

However in practice, there are major problems that impede the integration and consumption of LOD:

- **Incompleteness:** When consuming RDF data, meta information, such as ontological structures and exact range definitions of predicates, are desirable and are ideally provided by an ontology. However in the context of LOD and open RDF data, knowledge bases are often incomplete. This is especially the case for encyclopedical knowledge bases, such as Dbpedia [Bizer et al., 2009b], YAGO [Suchanek et al., 2007], or Freebase[1].

- **Inconsistency:** The incompleteness of knowledge bases is usually accompanied by inconsistency. Even when a knowledge base is available, we often observe triples that violate its axioms. Widely deployed ontologies often do not capture all aspects that are needed for domain-specific content of many data sets. Thus, data providers often define custom terms that are used in addition to vocabulary from widely deployed ontologies[2].

This inconsistency and lack of metadata impede the utilization of LOD. Inconsistency of data and meta information hinders automatic consumption of the data, nullifying the virtue of machine readability of data sources. Even manual consumption of the data by users becomes frustrating as data does not correspond to schema and ontology definitions. On top of that, interlinking and integrating such datasets is much harder, when data sources are inherently dirty and contain inconsistencies.

The first option to manage the mentioned quality issues is manual cleaning of data with the help of data publishers, ontology experts, and data analysts. However, manual rectification of data sources is cumbersome and can be applied only to very small datasets. To identify errors and misconceptions in large data sources, automated analyzing and profiling approaches are necessary. Furthermore, once misconceptions are identified, the decision on a reconciliation strategy depends on many factors. On the one hand, an ontology engineer needs to keep in mind which properties best describe an entity and on the other hand, she has

[1] `http://www.freebase.com/`

[2] `http://lod-cloud.net/state/#dereferencable-vocab`

to consider which properties users and data publishers actually expect when creating new records or looking up information. For example, to denote the weight of a person, DBpedia ontology provides the property `Person/weight`. However, for almost all person entities in the DBpedia dataset, the much simpler property `weight` is used, which actually is not defined in the ontology. In fact, hand crafted ontologies are too rigid to keep up with the increasing and changing volume of Lod.

Thus, it is useful to automatically generate meta information, such as ontological dependencies, range definitions, and topical associations of resources. In other words, the increasing amount of Linked Open Data (Lod) raises new opportunities and challenges for the data mining community [Heath and Bizer, 2011].

1.3 Data-driven Solutions

To scientifically position the contributions of this thesis, it is important to identify different streams of research that aim at the improvement of data quality and usage of open Rdf data, based on data-driven solutions. Several topics that contribute to the improvement of Lod and Rdf knowledge bases have so far been of academic interest in the Semantic Web and information retrieval community: Much research deals with the extraction of the data in the first place. Projects on general knowledge bases such as DBpedia [Bizer et al., 2009b] and YAGO [Suchanek et al., 2007; Hoffart et al., 2013] are prominent examples that deal with text extraction, natural language processing (NLP), and ontology matching. Other approaches such as [Lange et al., 2010; Wu and Weld, 2008] also used textual information to enrich the existing knowledge bases and ontologies. Textual resources have also been used in order to gather meta information that improves the ranking of Sparql query results [Kasneci et al., 2009c,b] or to identify possibilities for relaxing query results [Elbassuoni et al., 2011, 2012].

Another popular stream of research deals with the reconciliation of multiple sources. Well-known duplicate detecting techniques, such as the blocking and the sorted neighbourhood methods [Elmagarmid et al., 2007], were used to identify duplicate entities across different knowledge bases in order to connect them via `sameAs`-links [Volz et al., 2009; Böhm et al., 2012]. Schema matching approaches have also been adapted to match different ontologies [Suchanek et al., 2011].

Graph analysis and graph mining are also related fields, as each Rdf dataset can be considered as a graph where a predicate-edge connects a subject-node and an object-node. Graph analysis approaches, such as the random surfer model [Page et al., 1998], have been used for ranking entity relation ship data [Kasneci et al., 2009b; Langer et al., 2014], as well as for entity summarization [Cheng et al., 2011b], i.e., providing the most relevant facts of an entity.

In this thesis, we apply association rule mining to develop approaches that benefit the quality and the usage of Rdf data. From a methodological view, one can position the contributions of this thesis among data mining and probabilistic reasoning approaches that do not depend on external resources. Data mining and machine learning have been used for a limited set of applications that aim at improving Rdf data, mostly for entity-class relationship predictions [d'Amato et al., 2010; Fleischhacker et al., 2012] so far. In particular, association rule mining, although very popular in the information system community, was only marginally investigated on Rdf data and limited to domain-specific knowledge discovery scenarios [Nebot and Berlanga, 2010].

1.4 Contributions

In this thesis, we tackle the problems of incompleteness and inconsistency of open Rdf data from various perspectives. To improve the quality and the usability of Rdf data, we address the process of triple creation, ontology improvement, and query relaxation. Our algorithms are based on a methodology that adapts frequency analysis and association rule mining [Agrawal et al., 1993] to Rdf data. To this end, we make the following contributions:

Mining Configurations. There are many possibilities to apply association rule mining to Rdf data. We model the possibilities by proposing an approach that applies association rule mining at Rdf-statement level by presenting the concept of *mining configurations* [Abedjan and Naumann, 2011, 2013a]. A mining 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 configurations, we describe the corresponding opportunities and application fields.

Auto-Completion. To avoid inconsistency in data from the start, users should be supported during triple creation. 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 constitute 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). The creation of heterogeneous entries makes integration of the complete dataset difficult. Auto-completion through predicate suggestion remedies the problem, providing users with a list of commonly used predicates. We describe a rule-based solution to suggest properties as well as object values to an editing user. This work has been previously introduced in [Abedjan and Naumann, 2013a].

Triple Amendment. Applying the auto-completion approach to objects in addition to predicates enables us to also suggest object values as part of a new fact. We can then combine both approaches to *amend* data with completely new facts. We applied our mining concept to multiple datasets and achieved promising results in comparison to state-of-the-art systems. This work has also been briefly introduced in [Abedjan and Naumann, 2013a].

Ontology Alignment. While there are best practices for publishing Linked Open Data using established ontologies [Heath and Bizer, 2011], our analysis of the Billion Triple Challenge Dataset 2011 showed that for 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 and provide an algorithm that reconciles ontologies and data, following our rule-based methodology. The contributions of this chapter have partly been published in [Abedjan et al., 2012].

Synonym Discovery. Another possibility to deal with ontology misusage in RDF data is to relax query results. Browsing 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. This contribution has been introduced in [Abedjan and Naumann, 2013b].

The remainder of this thesis is organized as follows: First, we describe our mining methodology and compare it to state-of-the-art mining systems on RDF data. In Chapter 3, we show how rule mining can be applied for predicate/object suggestion and how both approaches can be extended for auto-amendment of new triples. In Chapter 4, we present our approach to deal with inconsistent

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

property usage with the help of ontology re-engineering suggestions. In Section 5, we show how to deal with the same problem of inconsistent ontologies via synonym analysis for predicate expansion. Chapter 6 describes our proof-of-concept tool ProLod++, which integrates the contributions of this thesis and has been presented as [Abedjan et al., 2014]. Finally, we conclude in Chapter 7 by summing up the results of this thesis and discussing open questions that may trigger further research on data mining in the Semantic Web.

Chapter 2

Mining RDF Data

In this chapter, we present and discuss our association rule mining methodology of mining configurations, which was briefly introduced in [Abedjan and Naumann, 2011]. First, we give a brief introduction to the concept of association rule mining and then we introduce our approach of mining configurations for RDF data. We will discuss semantics of basic configurations based on our experience on open datasets, such as DBpedia. Finally, we position our methodology among existing work and give an overview on data mining systems that deal with RDF data.

2.1 Association Rule Mining

The concept of association rules has been widely studied for transactional databases [Agrawal et al., 1993]. Conceptually, a transactional database consists of two main components: a set of transactions and corresponding transaction IDs (TID).

Each transaction t is a subset of an item universe I and and is identified with a unique TID. Regarding an e-commerce scenario, the item universe I would be the set containing all sold products. A transaction could represent all products purchased by a customer at a specific time (in a shopping basket). In that case, the TID would be a unique identifier composed by the date and customer ID. In a different transactional scenario a transaction could contain all the products a customer has ever purchased. In that case the customer ID would equal the TID. In such an e-commerce scenario, association rule mining aims at discovering correlations between items or item sets to analyse customer behavior and thus to improve product recommendations for future customers. Indeed, the formal definition of a transactional database is not restricted to any domain and can be applied whenever a great number of sets of re-occurring items are at hand. Before we present how RDF data can be appropriately transformed into a transactional database, we give a deeper view on association rules and algorithms that exist to obtain them from a transactional database.

2.1.1 Association Rules

An association rule is an implication $X \rightarrow Y$ consisting of the *itemsets* $X, Y \subset I$ with $X \cap Y = \emptyset$. Depending on the transactional database there can be an exponential number of itemsets $X \cup Y \subseteq I$ resulting in even more rules. Therefore, it is vital to remove all non-relevant options. To do so, the rule generation concentrates solely on *frequent patterns* (also called *frequent itemsetss* or *large itemsets*) $X \subseteq I$ that occur in a significant number of transactions.

The most popular measures for association rules that denote their relevance are *support* and *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 $X \cup Y$. Given a threshold minimum support *minSup*, all itemsets with a support above minimum support are called frequent patterns. 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. There are also other metrics for evaluating rules, such as lift and conviction [Brin et al., 1997], but for our approach support and confidence are sufficient.

While association rules usually denote correlating occurrence of items, *negative association rules* denote the degree of exclusive occurrence of items [Wu et al., 2004]. Negative association rules are association rules where the the absence of an itemset $X \subseteq I$ is denoted through a negation, e.g., $X \rightarrow \neg Y$ means that the presence of X implies the absence of Y. Similar semantics hold for $\neg X \rightarrow Y$, and $\neg X \rightarrow \neg Y$, or more complex rules such as $X_1 \neg X_2 \rightarrow \neg Y_1 \neg Y_2 Y_3$.

2.1.2 Algorithms

Given a set of transactions $T = \{t | t \subseteq I\}$, the core task for mining association rules is to count the occurrences of each item combination $i \subseteq I$ in all $t \in T$ in order to discover frequent patterns. Based on the retrieved frequencies it is possible to infer rules or other correlation statistics. Accordingly, algorithms to generate association rules decompose the problem into two separate steps [Agrawal and Srikant, 1994]:

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

While the second step of the algorithm is straightforward, the first step is the most time-consuming part of the rule mining task. The three best known approaches to this problem are Apriori [Agrawal and Srikant, 1994], FP-Growth [Han et al., 2000], and Eclat [Zaki, 2000]. For each of these algorithms, there exist multiple modifications and optimizations.

Apriori is based on a multi-path approach generating candidate frequent itemsets of a fixed size in each path, beginning with itemsets of size 1. The candidate generation allows a pre-pruning of infrequent itemsets for the actual support testing. It is based on the intuition that all subsets of a frequent itemset are also frequent itemsets. So candidate frequent itemsets of size k for each path are generated using the apriori known frequent itemsets of size $k-1$ from the previous path. There are also several optimized versions of the algorithms such as, DHP [Park et al., 1997] and RARM [Das et al., 2001]. FP-Growth is an algorithm that discovers frequent itemsets without a candidate generation step. It transforms the database into an extended prefix tree of frequent patterns (FP-tree). The FP-Growth algorithm traverses the tree and generates frequent itemsets by pattern-growth in a depth-first manner. Finally, the algorithm Eclat is based on intersecting transaction-id (TID) sets of associated itemsets, and is best suited for dealing with large frequent itemsets. Eclat's strategy for identifying frequent itemsets is similarily to Apriori based on candidate generation.

Discovering negative association rules is much more difficult as infrequent cannot be pruned anymore [Wu et al., 2004]. To efficiently discover positive as well as negative rules custom constraints have to be defined that prune the search space.

We use the FP-Growth algorithm. It is most appropriate for our intentions as the the prefix structure allows efficient retrieval of support values if we want to deliberately choose itemsets from the transaction database. We need that functionality when we want to identify the support of non-frequent itemsets to compute negative associations between frequent items that have the form $X \rightarrow \neg Y$.

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. For our mining approach, we directly use the subject-predicate-object (SPO) view of Rdf data, where information is represented as triples of resources and each triple represents an Rdf fact

Based on the SPO view, we choose one part of the statement as mining *context*, which is used for grouping one of the two remaining parts of the statement as the *target* for mining. Therefore, a transaction is a set of target elements associated with one context element that represents the transaction id (TID).

As a statement consists of three parts and any part of a statement can be either the target or the context of mining we have different opportunities to mine RDF data. We call each of those *context* and *target* combinations a *configuration*. Table 2.1 shows an overview of the six possible configurations and their identified use-cases. All six use cases refer to metadata generation: Schema analysis is very important in order to understand entities and their relations, while basket analysis where the term is borrowed from the traditional mining scenario provides information about the relationship of entities and their features. Clustering refers to identifying groups of similar entities in the data. In the next section, we further discuss the benefits and the difference of ontological clustering and topical clustering. Finally, range analysis and schema matching aim at identifying similar, redundant, or subsuming predicates. Each configuration can be further *constrained* to derive more refined configurations. For instance, the subjects may be restricted to the set of entities that are of type `Person`, or either the context or the target can be concatenations of two statement parts.

Table 2.1: Six configurations of context and target

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

In the following sections, we first introduce our notation that eases the illustration of different configurations. Then we further elaborate the meaning of the six configurations by highlighting the role of each statement part as the configuration context.

2.2.1 SPO-Notation

We represent a knowledge base as a set of triples $KB = \{(s,p,o)|s \in S \wedge p \in P \wedge o \in O\}$, where S corresponds to all resources that ever occurred as subjects of a statement, and P and O respectively contain all resources that ever occurred as the predicate or object of a statement. We use a raised letter to denote that a statement part is fixed by known values, e.g., ${}^{s}p^{o}$ denotes a predicate p connecting the given subject s with the object o and ${}^{sp}o$ the object value of the subject-predicate combination sp.

A capital letter always refers to a set of values, i.e., ${}^{s}P^{o}$ corresponds to the set of all predicates that connect s with o. Finally if one part of a statement is not of interest we just omit it, i.e., P^{o} represents all predicates that contain the object o in their range of values. Considering Table 2.2, the set $P^{Hamburg}$ contains the predicates `born` and `birthPlace`

Table 2.2: Facts in SPO structure from

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	occupation	Musician

2.2.2 Mining in the Context of Subjects

In the RDF model, all statements with the same subject represent one entity. Literally speaking, each triple is a fact about the triple's subject. While subjects represent entities in RDF data, predicates represent the schema for those entities. Configuration 1 represents a transactional database for mining predicates in the context of subjects. In that Configuration, subjects are TIDs, and each subject s identifies a transaction that corresponds to the set of predicate elements ${}^{s}P$ that occurred for the subject s in the dataset. Table 2.3a illustrates the set of transactions that correspond to our running example from Table 2.2. So, mining predicates in the context of subjects 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 triggers a closer examination of possible reasons for loose or tight consistency of the schemata, such as exisiting subcategries of entities or quality issues. Furthermore,

Table 2.3: Configuration examples

(a) Configuration: 1, Context: Subject, Target: Predicate

TID	transaction
Obama	*{birthPlace, party, occupation}*
Merkel	*{birthPlace, party, occupation}*
Brahms	*{born, occupation}*

(b) Configuration: 2, Context: Subject, Target: Object

TID	transaction
Obama	*{Honolulu, Democrats, Pres.}*
Merkel	*{Hamburg, CDU, Chancellor}*
Brahms	*{Hamburg, Musician}*

(c) Configuration: 3, Context: Predicate, Target: Subject

TID	transaction
birthPlace	*{Merkel, Obama}*
born	*{Brahms}*
orderInOffice	*{Merkel, Obama}*
occupation	*{Brahms}*
party	*{Merkel, Obama}*

(d) Configuration: 4, Context: Predicate, Target: Object

TID	transaction
birthPlace	*{Hamburg, Honolulu}*
born	*{Hamburg}*
orderInOffice	*{President, Chancellor}*
occupation	*{Musician}*
party	*{CDU, Democrats}*

(e) Configuration: 5, Context: Object, Target: Subject

TID	transaction
Musician	*{Brahms}*
Hamburg	*{Brahms, Merkel}*
Hawaii	*{Obama}*
President	*{Obama}*
Chancellor	*{Merkel}*

(f) Configuration: 6, Context: Object, Target: Predicate

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

these correlations can be used to identify anomalies such as missing facts. Based on this intuition, we design a rule-based approach for auto-completion and auto-amendment of Rdf data in Chapter 3.

Having frequencies of patterns at hand, it is also possible to consider negative correlations. Considering negative correlations among predicates hints at the exclusive occurrence of predicates. On the one hand this exclusivity might endorse the fact that the predicates describe completely different type of entities. For example, the two predicates `birthPlace` and `foundationPlace` that describe properties of different types such as `Person` and `Organisation` occur only exclusively, i.e., the itemset `{birthPlace, writer}` has a support of 0% while both predicates are frequent items with a support above minimum support. Identifying predicates of this sort enables to use them as features for clustering algorithms. Another intuition is that if those predicates occur for entities of the same type, they might have the same meaning and are therefore used alternately. We elaborate this intuiton in Chapter 5, when we present our approach on predicate expansion.

In accordance with our view of entities and schemata, objects would represent the actual values that describe an entity. Thus, when mining objects in the context of subjects (Configuration 2), each transaction corresponds to all entities or literals ^{s}O that occurred as objects for the subjects s. Likewise, mining in the context of subjects is the process to discover patterns between values that are associated with each other by co-occurring for many entities (see Table 2.3b). For example, the rule `Buenos Aires` $\rightarrow$ `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 such strong rules.

Having a fixed set of transactions, denoted by all distinct subjects S, the probability to have more repeating predicates among the subjects is higher than for objects. As a matter of fact, an RDF knowledge base usually contains more distinct objects than distinct predicates. For example, DBpedia v. 3.8 contains 1,313 distinct predicates and 5,172,511 distinct objects. Therefore the support thresholds for Configuration 1 and 2 have to be respectively adapted, reflecting the ratio of the number of distinct predicates to the number of distinct objects, to obtain any frequent patterns in the first place.

2.2.3 Mining in the Context of Predicates

To mine in the context of predicates means that we have a transaction base where each predicate identifies a transaction. Regarding the basic configuration methodologies, Configurations 3 and 4 correspond to such a transaction base. When mining subjects in the context of predicates each predicate p is associated with a transaction $p \in S^{p}$ containing all subjects that occur with p in the dataset. Frequent subjects in such a transaction database correspond to subjects with many common predicates. From an entity relationship perspective, subjects with common predicates have similar schemata which means the entities belong to a specific class of entities. 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` $\rightarrow$ `Lyndon B. Johnson` with 92% confidence. As rules arise 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 more rules that connect also athletes and presidents, because they share predicates of the more general concept `Person`, such as `name` or `birthPlace`. Regarding our running example in Table 2.3c, we observe that Obama and Merkel co-occur for multiple predicates. Obviously, both entities have a similar occupation and represent politicians.

When mining objects in the context of predicates, each transaction ^{p}O comprises the actual range[1] of the specific predicate p. Exemplary rules from DBpedia include `1` $\rightarrow$ `{2, 3}` or `Albania` $\rightarrow$ `Italy`. In fact, the mining results of this configuration is very similar to the configuration for mining subjects in the context of predicates. However there are two differences. First, Configuration 3 discovers rules among literals, such as numbers and strings, too. Second, each transaction already represents a class of entities, as the range of a predicate is usually implicitly well-defined, e.g., `birthPlace` has only locations in its range. Re-occurring patterns denote attribute values that are used in a wide range of domains. For example, country names co-occur as frequent patterns, because these can be attribute values from different perspectives, such as `birthPlace`, `deathPlace`, `foundationPlace`, `residence`, etc. Also certain numbers can be frequent patterns occurring for dates, years, number of wins/losses, etc. Our running example in Table 2.3d denotes the co-occurrence of the locations `Hamburg` and `Hawaii`.

Both configurations in the context of predicates are similar in the sense that on one hand there are only a small number of transactions and on the other hand these transactions contain a large amount of items. In case of the DBpedia v. 3.8 dataset, we have 1,313 transactions with on average 12,309 items for Configuration 3 (mining target: subjects) and 4,839 items for Configuration 4 (mining target: object), while for example in Configuration 3's reverse Configuration 1 (mining context: subjects) we have 2,342,853 transactions with 6 items on average. A possibility for reducing the number of items in a transaction is to consider more abstract concepts that contain the items, such as data types or affiliated classes.

2.2.4 Mining in the Context of Objects

Similar to previously described configurations, mining in the context of objects denotes a configuration, where object values of an RDF dataset denote transaction identifiers while predicates or subjects constitute the items to be mined.

Thus, when mining subjects in the context of objects (Configuration 5) each transaction S^{o} denotes the set of all resource identifiers that occur as subjects in the same facts as the object o in the underlying dataset. Objects are values that might be associated with subjects in different relations. For example, several entities of type `Person` may share the object `Berlin` in different roles like `birth` or

[1]In some knowledge bases, such as the DBpedia ontology, ranges and domains of predicates have been identified by one or more ontological classes. However, an actual dataset might contain facts that are not captured by such a definition.

`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. Thus, discovering frequent patterns in this scenario is the process to discover subjects that share a minimum number of object values, which in turn results in rules between entities that are topically related.

When mining predicates in the context of objects as denoted by Configuration 6, each transaction P^o denotes the set of predicates that have the object o in their range of values. Therefore, frequent predicate patterns in the context of objects are set of predicates that have a high 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. In Chapter 5, we describe how we apply Configuration 6 to discover synonymously used predicates.

So far, we presented mining configurations- our methodology for mining RDF data. In particular, we conjugated each basic configuration and its presumable semantics and area of application. In the next section, we position our methodology among existing work by giving an overview of approaches on mining RDF data and comparing existing approaches to our approach.

2.3 Related Work

Categorically, our methodology belongs to the research field of Semantic Web mining. However, Semantic Web mining is a very broad and diverse research field that includes semantic approaches to mine the web as well as mining approaches applied to the Semantic Web [Stumme et al., 2006]. We limit the scope of the related work based on the input for the mining systems, i.e., we focus on the approaches that consume RDF or similarly structured data. To this end, we identify the three not necessarily disjoint subtopics, association rule mining, ontology learning, and graph mining.

2.3.1 Association Rule Mining

Association rule mining on RDF data is an emerging topic with several new use cases. Initially, rule mining on RDF data was rather a supportive technique for exploratory analysis of a given knowledge base. Our profiling tool ProLOD provides a rule mining engine for identifying positive and negative rules between predicates [Böhm et al., 2010]. Indeed, ProLOD's rule mining engine directly corresponds (but is also limited) to Configuration 1 of our mining configuration methodology. Putting predicate mining into use, Joshi et al. presented a rule based approach to compress RDF data [Joshi et al., 2013].

Nebot et al. present a rule mining framework that uses association rules to infer knowledge in domain-specific datasets [Nebot and Berlanga, 2010]. They present a SPARQL framework in medical RDF data to discover drug and disease correlations among patients. In their framework, one can deliberately choose an entity type as the mining context, e.g., `Patient`, and another type as mining targets, e.g., `Disease` and `Drug`. Considering the predicates in the dataset as directed edges of a graph, a transaction then contains all mining targets that are reachable from one context entity. In this way they can mine rules between diseases and drugs, considering each patient as a transaction. This model is appropriate for domain-specific scenarios where domain experts already have profound knowledge about the data or instances are consistently typed according to an ontology. Our model is more generic and can also capture these scenarios by constraining certain configurations.

Another stream of works concentrates on rule-based or statistical methods for ontology learning [Parundekar et al., 2010; Völker and Niepert, 2011; Fleischhacker et al., 2012]. A crucial step for these methods is to semi-automatically acquire a desired terminology, such as classes and properties. This terminology then comprises the items of transactions upon which a rule mining approach can discover subclass relationships or range restrictions [Völker and Niepert, 2011]. An extended version of this approach is also able to discover inverse predicates [Fleischhacker et al., 2012]. Again, our methodology is much more abstract as we only consider the SPO semantics that subsumes scenarios were the set of predicates and objects are restricted. In Chapter 4, we discuss this stream of approaches again considering the actual use case that is related to our ontology improvement approach.

A recent system for association rule mining in RDF data is AMIE [Galàrraga et al., 2013], which is inspired by WARMR [Dehaspe and Toivonen, 1999] and [Józefowska et al., 2010] that use a declarative language to mine association rules on small sets of conjunctive queries. The basic idea of AMIE is to concentrate on mining horn rules among relations such as

$$\textit{hasChild(p,c)} \land \textit{isCitizenOf(p,s)} \rightarrow \textit{isCitizenOf(c,s)}$$

, where a triple *spo* is denoted as a relation *p(s,o)*.

Such a horn rule contains a conjunction of relations in its condition part (the body) but only one relation in the consequent part (the head). The interestingness of such a rule can also be measured via support and confidence, which can be obtained by considering the number of instantiations of bound variables within the relations. However, a major difference of this methodology to our's is that there is no transactional database. As a result minimum support cannot generally be determined for a mining task. Therefore, the creators of AMIE presented their own relevance measure: head coverage, which is the ratio of the instantiations of all relations in a rule and the number of instantiations of the rule's consequent. The authors further propose a new confidence method which will be discussed in more detail Chapter 3, where we also present experimental comparisons with AMIE.

2.3.2 Ontology Learning and Reasoning

A significant amount of related work on machine learning in the Semantic Web originates from inductive logic programming (ILP), which bridges reasoning techniques and statistics [Muggleton, 1995]. Similar to the horn rule methodology presented in the previous section, ILP concentrates on entailing relations by testing them on a knowledge base. Based on a general reference concept, additional logical relations are considered for refining the entries that fulfill a conjunction of relations. ALEPH [Muggleton, 1995], and Sherlock [Schoenmackers et al., 2010] are known systems to mine such rules. ALEPH is an ILP system based on Muggleton's Inverse Entailment Algorithm [Muggleton, 1995]. Sherlock uses a probabilistic graphical model to infer first order clauses from a set of facts for a given relation [Schoenmackers et al., 2010]. Lisi et al. also present an approach to mine rules on ontologies and datalog programs [Lisi and Esposito, 2005]. These approaches depend on a clean ontological knowledge base without factual errors, which is usually not available.

Complementing the ILP method, plenty of machine learning systems such as, similarity-based class-membership predictions, kernel based methods, and multivariate prediction models have been introduced, as presented by Rettinger et al. [Rettinger et al., 2012]. Here we point out some representatives of these machine learning systems to show how RDF data is consumed by this sort of algorithms. In following chapters, we will again refer to some of the approaches that are related to our use cases.

As pointed out in Section 2.2.2, features for machine learning tasks could be obtained via pattern analysis in the context of subjects. Identifying predicates and object values of entities as discriminating features, d'Amato et al. present an approach to predict class membership of individual entities [d'Amato et al., 2008] or to do unsupervised clustering [Fanizzi et al., 2008]. Further research on that topic includes machine learning based on neuronal networks [Fanizzi et al., 2009], and multivariate prediction [Rettinger et al., 2009]. Recently, D'Amato et al. proposed approaches to enrich ontologies applying ILP to heterogeneous sources, such as RDBMS and web sources [d'Amato et al., 2012, 2010].

2.3.3 Graph Mining

As RDF data spans a graph, where resources (subjects and objects) represent the vertices and are connected by predicates as edges, another related field of research is mining frequent subgraphs or subtrees [Chi et al., 2004; Kuramochi and Karypis, 2001]. 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.

Some mining techniques underly the random surfer model PageRank [Page et al., 1998]. The MING algorithm [Kasneci et al., 2009a] introduces an informativeness measure that builds on a natural extension of the random surfer model in order to rank RDF query results based on their informativeness. RELIN [Cheng et al., 2011a] is an entity summarization approach that is based on the random surfer model. In fact, entity summarization can be regarded as a related field, as the idea is to identify the most interesting set of properties to describe an entity. Yet, considering only the frequency of properties might be misleading as some properties, such as `rdfs:type` or `rdfs:label`, are frequent but not necessarily informative. Further approaches that focus on that topic includes a system, which is based on kernel graphs [Thor et al., 2011] and the recent system DIVERSUM [Sydow et al., 2013] that focuses on diversification in graphical entity summarization.

2.4 Summary

Association rule mining is a traditional unsupervised approach to identify frequent patterns and correlations of itemsets within a transactional database. To apply association rule mining on RDF data, we introduced the concept of mining configurations where one part of an SPO triple can be regarded as the context of mining and another as the target of mining. Each configuration harbors implicit semantics that suggest specific use cases. Previous research on association rule mining on RDF data was limited to horn rule mining or to specific domains, where data was arbitrarily transformed to a transactional database based on a specific use case. Our approach differs from existing mining approaches in a sense that our model is a generalization of the possibilities to mine RDF data.

In the following chapters, we present three different use cases that put individual configurations from our methodology into use. In particular, we show how the combination of more than one configuration enables us to infer useful knowledge from RDF knowledge bases.

Chapter 3

Enriching RDF Data

Consistency and completeness are crucial aspects regarding the quality of LOD knowledge bases. Both aspects are directly affected by the creation process of such a knowledge base. Knowledge bases that are directly derived from structured data, usually have a high degree of consistency in their property usage. However, the majority of data sources that have been extracted from unstructured sources, such as DBpedia, suffer from redundancies and inconsistencies that are caused by flaws in the extraction process as well as semantic interpretation of extracted data[1]. Additionally, some datasets, especially encyclopaedical datasets, undergo periodical changes, because the underlying data (e.g. Wikipedia Infoboxes[2]) evolves in a collaborative manner.

When creating new triples manually, one would expect that the creator exactly knows which properties and values should be created. However, regarding existing LOD data sets this is clearly not true. For instance, editors of Wikipedia infoboxes 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` or `weight` instead of `Person/weight`). Those heterogeneous entries make integration of the complete dataset difficult. Furthermore, a new user might spend too much time to identify appropriate property elements for creating a new entry. Especially in such situations it is important to provide mechanisms that maintain some degree of consistency when data is updated.

Some would argue that providing a well-defined ontology and a template remedies such issues. However, in reality there is a clear divergence with regard to the conformance of LOD and underlying ontologies [Abedjan et al., 2012]. Reality is too complex to be covered by fixed static templates, and schema drift occurs. In

[1] `http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/DataSets`

[2] `http://en.wikipedia.org/wiki/Help:Infobox`

case of Wikipedia infoboxes one could imagine to use the appropriate infobox-template[1] for suggestions. Users often ignore provided templates and use their own custom properties that they judge to be more semantically fitting. While the template `Infobox company` asks for a `name`, the vast majority of company infoboxes uses `companyName` instead. Furthermore, the new WikiData project[2] for building up a global knowledge base is trying to provide user editing support without having schema or template information.

Therefore, we argue that an instance-based approach for suggesting new fact entries based on predicate mining is better suited to solve this task. On the one hand, the suggestion systems maintains consistency by suggesting commonly used predicates based on existing entities, and on the other hand, the data-driven aspect of the solution provides more intuitive suggestions based on collaboratively confirmed properties avoiding semantical insufficiencies of obsolete property definitions.

Once it is possible to infer property usage based on data mining, it is obvious to apply the suggestion system for remaining parts of a statement, too. Especially as we presented a general concept for data mining based on context and target configurations, it is technically possible to adapt mining system for suggesting predicates also to objects. With the approach applied to objects, we can combine both approaches for *amending data* with completely new facts about a given subject.

In this chapter, we first discuss related work on fact and link generation. Then, we show the basic intuition behind predicate suggestion and fact amendment, and how both approaches are related. In particular, we directly apply our methodology on mining configurations in the context of subjects and with predicates or objects as mining targets (Configurations 1 and 2 from Table 2.1). The intuitions of our approach were already introduced in [Abedjan and Naumann, 2013a]. In Section 3.3 and Section 3.4, we elaborate the technical details and algorithms for auto-completion during triple creation and auto-amendment of new triples, respectively. In Section 3.5, we present a large number of experiments confirming our claims on data-driven enrichment of RDF data. Finally we summerize the key insights in Section 3.6

3.1 Related Work

Generation of new facts and suggestion of statement parts are related to several research fields, such as reasoning, rule based ontology creation, ontology learning, link creation, and schema auto-completion. Some of the approaches, especially

[1]`http://en.wikipedia.org/wiki/Template:Infobox`
[2]`http://www.wikidata.org/wiki/Wikidata:Main_Page`

those on association rule mining have already been discussed with regard to the methodological application of rule mining in the previous chapter. In this chapter, we rather focus on the specific use case of these systems that aims at enriching RDF data.

3.1.1 Reasoning

Since the very beginning of the Semantic Web, reasoning was one of the most relevant research fields. Given a set of facts or axioms, reasoning aims at inference of logical consequences based on description logics to enrich the set of axioms or to create richer ontology definitions [Baader et al., 2005]. By extending standards such as RDF and RDFS to the DL-based ontology language OWL [Horrocks et al., 2003], multiple reasoning approaches were published [Urbani et al., 2010; Mutharaju, 2012].

Many efforts have been put into fuzzy reasoning techniques to deal with uncertainty and imprecision of data [Stoilos et al., 2005, 2007]. Nevertheless, logical reasoning faces two major challenges: First of all, the number of inferred facts is usually too large to be verified. Second, existing data inconsistencies lead to the inference of faulty statements [Bonatti et al., 2011; Hogan et al., 2009]. Data inconsistencies and misusage of ontology axioms make it nearly impossible to infer new knowledge based on given axioms. Our approach is fully statistics based and does not need to rely on manually crafted vocabularies, such as logic and ontology descriptions.

3.1.2 Rule-based fact generation

Regarding ILP (Inductive Logic Programming) systems, ALEPH [Muggleton, 1995], WARMR [Dehaspe and Toivonen, 1999], and Sherlock [Schoenmackers et al., 2010], and AMIE [Galàrraga et al., 2013] are known systems that generate new facts based on a given knowledge base, as presented in Section 2.3. In a number of experiments, AMIE showed to be the most efficient and effective approach to generate new facts compared to ALEPH and WARMR [Galàrraga et al., 2013]. Therefore, we compare our system to AMIE. As previously discussed, AMIE concentrates on mining horn rules among relations, such as

$$\mathit{hasChild(p,c)} \wedge \mathit{isCitizenOf(p,s)} \rightarrow \mathit{isCitizenOf(c,s)}$$

Based on support and confidence thresholds on the instantiations of the variable subjects and objects, such a rule entails new relations *isCitizenOf(c,s)*. The generation of these horn rules follows specific constraints, in order to reduce the number of generated rules. In particular, each rule has to be closed, i.e., each of the variables in the relations should be shared in at least one other relation.

A further notable contribution of AMIE over previous work is the introduction of the *partial completeness assumption (PCA)*, which assumes that if a dataset contains some facts on a subject s with a predicate p, it knows all the possible facts of that subject with the predicate p. Based on the partial completeness assumption, they present the PCA-confidence that normalizes the number of instantiations of the head atom of the rule by all possible instantiations that also include different objects than the head atom.

Statistical schema induction is also a related field, as induction of schemata describing facts, such as class-membership or domain and range definitions, can be considered as part of an ontology [Völker and Niepert, 2011]. However Völker and Niepert's approach already assumes that some terminology from the dataset is known. Fleischhacker et al. extended their approach to discover inverse relations [Fleischhacker et al., 2012].

Given that rules in RDF data imply the existence of new facts, they can be regarded as compression dictionary elements for RDF data. Joshi et al's approach on compressing RDF data basically follows the same intuition as our approach but in the reverse direction: Their approach compresses data by removing facts that can be generated through association rules [Joshi et al., 2013].

In Section 2.3, we already introduced a set of machine learning algorithms that aim at class-membership predictions and that again can be regarded as approaches that enrich a knowledge base [Rettinger et al., 2012].

Another stream of approaches tries to populate knowledge bases by using additional data sources. D'Amato et al. propose approaches to enrich ontologies applying ILP to heterogeneous sources, such as RDBMS and web sources [d'Amato et al., 2012, 2010]. In the field of text extraction, iPopulator [Lange et al., 2010] and KYLIN [Wu and Weld, 2007] concentrate on populating Wikipedia infoboxes with new facts, using the Wikipedia text articles and existing infobox templates for training. Our approach does not rely on external data sources or structural information, such as ontologies, templates, or texts, but only on the existing RDF statements of the current RDF corpus.

3.1.3 Linking across data sources

Considering the complete LOD cloud, interlinking data sources in the spirit of entity matching via `sameAs`-links can also be considered as a field of research related to enriching RDF data [Ding et al., 2010]. One of the most notable systems in that area is Silk that links entities based on heuristics and manually chosen similarity metrics [Volz et al., 2009]. This approach has been further improved by applying blocking strategies [Isele et al., 2011] and genetic algorithms [Isele and Bizer, 2012]. Finally, LINDA is distributed link discovery approach based on hadoop, which scales for very large datasets [Böhm et al., 2012].

Another stream of research deals with holistic alignment of both entities and schemata, such as RiMOM [Li et al., 2009], Illiads [Udrea et al., 2007], and Paris [Suchanek et al., 2011]. RiMOM is a framework that automatically selects matching strategies to match instances as well as schemata. Illiads additionally combines logical reasoning with similarity based matching. Paris is a probabilistic algorithm for aligning instances as well as schema definitions among ontologies [Suchanek et al., 2011], which in particular follows the same research line as AMIE, because both consider the functionality degree of predicates.

Our fact generation system differs from the link generation and ontology alignment algorithms with regard to two aspects: First, in contrast to our mining configuration system, none of the mentioned approaches in this field is designed to generate links within a single dataset. Second our system is based on rule mining and does not rely on property similarities, although taking similarities and fuzzy mining into account might be an interesting field for future research.

3.1.4 Auto-Completion

Suggestion and auto-completion systems have already been subject of research in different areas. A similar use case has been applied by the authors of [Cafarella et al., 2008] in the context of web-scale extraction of structured data for knowledge base creation. They introduce a system that helps database designers to create schemata by providing schema auto-completion based on extracted web tables. As their approach is based on schema frequencies and probabilities, the methodology is quite similar to our approach for predicate and object suggestion. Their suggestion scenario is based on fully structured data, where the schema for relational tables is suggested by auto-completion. In addition to the properties of an entity, we also propose value entries for the schema elements. Finally, our target user for the suggestion and data amendment is not the ontology designer nor in their case the database designer, but the end-user who creates new data content.

Another similar field for auto-completion that benefits user-interaction and data consolidation is tag-recommendation. In [Menezes et al., 2010] the authors present a recommendation algorithm based on association rules. Tag-recommendation supports annotating users of websites or tweets, by recommending new tags, based on already entered tags. The authors report better scalability than algorithms that are based on decision trees and other similar classifiers. Our methodology allows to mine and combine very different facets of Rdf statements, while tag mining would technically correspond to one single mining scenario. Furthermore, we introduce a new approach for creating semantically valid facts by combining predicate and object mining. Our experiments show the capabilities of association mining on Rdf data and the fact that association rule mining leads to better results on predicate suggestion compared to tag recommendation scenarios. Furthermore, creation of complete new facts is more complex than single value recommendations. To this end, we are not aware of any RDF suggestion system that aims at supporting knowledge base editors with suggested properties or values.

3.2 Roadmap for Enriching RDF Data

In shopping basket analysis, one application for association rules is cross-marketing and product re-organization. Having a rule, such as *milk* → *cornflakes*, the company considers changing its marketing strategy to sell *cornflakes* to all of the customers who buy milk and to leverage the confidence for the rule. The intuition is that customers who exclusively bought milk either forgot to buy cornflakes or are at least willing to buy them when offered. The same intuition can be applied to the process of knowledge creation, as one can assume that some entries that are highly correlated with other entries might be forgotten. We distinguish two different scenarios for enriching RDF data (see Fig. 3.1):

1. *Auto-completion* of new fact entries through predicate or object value suggestion for the given subject.
2. *Amendment* of RDF data with completely new triples, based on patterns over existing triples and the previous suggestion step.

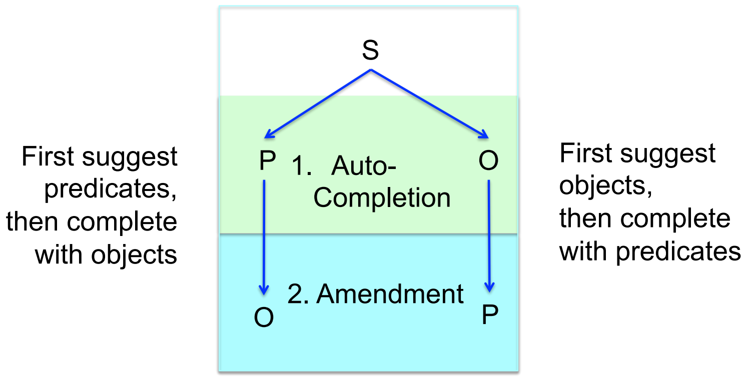

S
First suggest predicates, then complete with objects
P
1. Auto-Completion
O
First suggest objects, then complete with predicates
2. Amendment
O
P

Figure 3.1: Two paths for the suggestion of objects and predicates given a subject

Figure 3.1 visualizes the alternative options for enriching RDF statements. Both paths require two steps. Given a subject, the first step represents the suggestion scenario where either an object or a predicate can be suggested. In the second step based on the given subject and the suggested predicate/object from the step before, the missing part has to be selected for amendment of one triple. Unlike related approaches we do not rely on external knowledge, such as ontologies or textual information. In the following, we elaborate both paths and present our rule-based approach for schema and value suggestions for new knowledge base entries. Then we explain how we derive an approach for automatic fact amendment.

3.3 Auto-Completion

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 and is encouraged to contribute more entries to a dataset. Second, system feedback might prevent the user from using inappropriate synonyms for predicates as well as objects. In the following, we discuss both issues, comparing predicate suggestion with object suggestion.

3.3.1 Suggesting new Predicates

The suggestion workflow for both predicates and objects is identical, with the difference that the targets of suggestion and also of mining are either predicates or objects respectively. Referring to our classification in Sec. 2.2.2, our mining scenario corresponds to Configuration 1 when suggesting predicates and to Configuration 2 when suggesting objects. 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. All algorithms presented in Section 2.1 are valid options for this step; we used the FP-Growth algorithm [Han et al., 2000].
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 the other index the consequents of a predicate 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 the user is inserting or editing facts related to a specific subject s from a knowledge base $KB = \{(s,p,o)|s \in S \wedge p \in P \wedge o \in O\}$, the system can look up all the predicates that have already been inserted for the current subject ${}^{s}P_0 = \{p|(s,p,*) \in KB\}$. Based on ${}^{s}P_0$ we are able to compute the set of predicates ${}^{s}P'_0$ out of the set of all predicates P that are to be suggested for the current subject s, using the following formula:

$$^sP_0' = \{p \in P | aConf(^sP_0, p) \geq minConf\}$$

The set $^sP_0'$ contains all predicates from P, for which the function *aConf* exceeds the minimum confidence threshold *minConf*. Here, *aConf* aggregates the confidence values of all available rules $conf(Q \rightarrow p)$ with $Q \subseteq {}^sP_0$ and creates one overall confidence value between 0 and 1. In our approach, we took the sum of all squared confidence values and normalized it by dividing by the number of schema elements in $^sP_0$:

$$aConf(^sP_0, p) = \frac{\sum_{Q \in {}^sP_0} conf(Q \rightarrow p)^2}{|^sP_0|}$$

Taking the sum of all squared confidence values ensures that the occurrence of few high confidence rules has more impact than the occurrence of many low confidence rules. We also considered rules between sets of predicates, however we could not achieve any performance improvement. Therefore, we limited our approach to rules between distinct predicates for runtime efficiency.

Having computed the set $^sP_0'$, which contains all predicates with aggregated confidence above the given threshold, the results can be sorted by their aggregated confidence values and presented to the user. When the user chooses the next predicate p to insert into the data set, $^sP_1'$ has to be recomputed based on the new schema set $^sP_1 = {}^sP_0 \cup \{p\}$. Theoretically, it would also be necessary to recompute all association rules, however the effect of one added predicate is insignificant with regard to the large amount of data.

Regarding our example from Tab. 2.2, imagine we are to insert a record for *D. Cameron* by beginning with the statements *"Cameron birthPlace London."* and *"Cameron orderInOffice Prime Minister."*

That would result in $^sP_0 = \{$`birthPlace, orderInOffice`$\}$ and the total set of remaining predicates would be $P = \{$`party, instrument`$\}$. Considering only rules of size 2, the set of predicate rules relevant for the next suggestion include

1. `birthPlace` →`party` with 66.7% confidence.
2. `orderInOffice` → `party` with 100% confidence.
3. `birthPlace` → `instrument` with 33.3% confidence

Having *minConf* = 50%, the predicate `party` would be added to P_0' because, $aConf(^sP_0, party)$ is above *minConf*:

$$aConf(^sP_0, party) = \frac{1^2 + 0.67^2}{2} = 0.72 > 0.5$$

3.3.2 Value Suggestion

The suggestion of objects is technically equivalent to the suggestion of predicates, but the amount of objects is by magnitudes larger resulting in weaker rules. For instance, the DBpedia 3.6 data set contains 1,100 distinct predicates but 3,980,642 distinct objects. Furthermore, for a user, choosing an object value for a proposed predicate is expected to be more convenient than vice versa, because an object value can be looked up in general encyclopaedias while properties are defined manually with semantics that are unknown to the user. For example, a user might have created the entry:

`B. Obama birthPlace Honolulu.`

Following the object suggestion, 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 might not even know how the subject and the proposed object are connected and which predicate (`birthPlace`, `residence`, etc.) to choose. In addition to the semantical fitting of the predicate, the user also has to consider its appropriateness with regard to consistency among similar entities. Ontologically similar entities should share the same predicates. Nevertheless, once the user decides to enrich his current entity with either a new object or predicate, the suggestion of the respectively remaining part will be eased, because two parts of a statement are already known. In the next section, we elaborate this thought and show how the two suggestion scenarios work when combined.

3.4 Fact Amendment

When considering the generation of completely new statements, it is necessary to realize that we need an initial point for enriching the data. In other words, we need to decide which resource has to be extended. Therefore, at some point one value, either a subject, a predicate, or an object must be selected for being extended with new values for the two other missing parts. Regarding Fig. 3.1 that outlines a roadmap for suggestion and auto-amendment, we might already know subjects that might be extended with new values.

The roadmap of Figure 3.1 implies that after we were able to suggest one missing part for a given subject, it is possible to also complete the remaining third part. For example, if the system decides to suggest the predicate `residence` for *B. Obama* it is also able to choose the right object, e.g., *Washington D.C.* from the existing value range of `residence`. Vice versa, if the system follows the object suggestion path and suggests the object *USA* it should be able to choose also the appropriate predicate. 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*.

A different way of creating new statements is to let the system itself choose the subjects that should be amended with new triples. We call this approach *data-driven auto-amendment*. In the following we discuss both approaches and describe how we derive the data-driven approach from them.

3.4.1 User-driven auto-amendment

The step from suggestion to auto-amendment requires the application of rule mining for the respectively remaining configuration in the context of subjects. Depending on the choice of the statement part in the suggestion step, in the completion phase the remaining part can be selected on the basis of the subject and the suggested part.

Object Completion

Now consider the first path for amending a new statement from Figure 3.1. Given the fact that the predicate is suggested for a subject, the object value for the complete triple amendment would be selected in the next phase. Again, we need all association rules involving objects ${}^{s}O = \{o|(s, *, o) \in KB\}$ from the existing facts of the current subject s. For now, the completion is quite similar to the suggestion scenario. The major difference is that we now additionally know the predicate ${}^{s}p$ that has to link the current subject with one object. Knowing the predicate, we can prune all objects that have never been in the range of this predicate within the data set and consider only the rules that have the objects ${}^{p}O = \{o|(*, p, o) \in KB\}$ in their consequence.

For example, given the subject *B. Obama*, object rules might lead to completion candidates, such as *USA* or *Christian*, but knowing that the predicate is `residence`, the candidate *Christian* would be ignored, as it is not in the range of `residence`.

Predicate Completion

Considering the path that is based on object suggestion in the first place and a following predicate completion from Figure 3.1, we can analogously apply the same methodology introduced for suggestion by interchanging the role of predicates and objects. The completion step benefits from the fact that it can already remove predicates from the suggestion set that have never had the currently suggested object in their range.

Comparison of the completion options

Regardless of the fact, which of the two user-driven auto-amendment paths from Fig. 3.1 is chosen, one always faces the difficulty of selecting the correct object among a large set of possible solutions. However as experiments in Section 3.5 will show, the path that starts with object suggestion is more promising. This is due to the fact that a suggested object allows only a few predicates to be selected in the amendment phase, while the number of objects in the range of a suggested predicate is up to 100 times higher. Having a user in the middle who accepts or reject suggestions of the first step, the completion step can work quite accurate.

In a fully automated process however, we can simulate the users confidence by having a more cautious confidence threshold. By considering only high-confidence rules (e.g., above 90%) for object suggestions one can filter a large amount of possible new facts in beforehand. The obvious drawback of such an intuition is that the required existence of high-confidence object rules for each entity that is being edited by a user. In the following, we show how to deal with this dilemma following a data-driven approach.

3.4.2 Data-driven auto-amendment

We stated that object suggestion faces the problem of uncertainty. However, when allowing only suggestions based on high-confidence object rules the uncertainty sinks dramatically. By regarding only high confidence rules among objects, we are able to filter a large amount of unnecessary object choices. Furthermore, we achieve higher certainty by having the much smaller set of predicates in the completion step. In this data-driven approach the subject to be amended with a new fact is selected on the basis of existing high-confidence object rules.

Algorithm Intuition

Our approach is based on the following intuitions, which constitute two consecutive steps of the overall algorithm:

Expanding high-confidence rules. For object rules $O' \rightarrow o$ with high confidence (above 90%) and $O' \subseteq O$, the subjects $S^{O'}$ occurring with the objects O' are also likely to occur with the object o. However, up to 10% of the subjects that occur with O' *violate* the rule by not occurring with o in any fact. Those facts may be absent, either because they resemble exceptions to the rule or 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. However someone querying the graph for people born in the USA will miss this specific entity where only the information about Honolulu is available.

Rejecting facts without predicate support. A subject s should not be enriched with a fact containing object o if on the basis of the rules involving schema predicates ${}^{s}P$, no predicate can be chosen for the connection with o. This intuition allows a softening of the earlier intuition that expects all subjects that violate $O' \rightarrow o$ should be extended with a triple containing o. If no proper predicate can be chosen, we conclude that the subject should not be enriched with a new fact containing o_2.

Theoretically, one could adapt the intuitions based on high confidence predicate rules. However, the discovery of the appropriate object for a to-be-added predicate is much more cumbersome, because of the large amount of available objects. Concerning the first intuition one could argue that some of these implicitly given facts may be generated using ontological dependencies within the data. However, only relying on ontological dependencies results into many errors as soon as a few faulty facts exist. To avoid such a discrepancy it is important to apply statistical reasoning as we do in our approach with our frequency-based rules. Furthermore, not all implicit dependencies in the real world are captured within an ontology. For example the high-confidence object rule *South Park* $\rightarrow$ *Trey Parker* among television episodes correctly suggests that Trey Parker is involved in all episodes of South Park and should be added as the producer when absent in the data. While an instance-based association rule might hold for this TV show, there cannot be a general ontological rule that each episode of a series should have the same producer as listed for the complete series entity.

Algorithm Overview. The algorithm for data-driven auto-amendment is divided into four steps:

1. Create predicate-predicate rule matrix in the context of subjects as described in Section 3.3.1 (Configuration 1 in Table 2.1).

2. Generate high-confidence object rules $o_1, o_2, \ldots o_n \rightarrow o$ (Configuration 2 in Table 2.1).

3. Exclude all rules that are redundant. That means, we remove all rules $O' \rightarrow o$, if there is a more general rule $O'' \rightarrow o$ with $O' \supset O''$, because $S^{O''} - S^{o}$ contains all subjects from $S^{O'} - S^{o}$.

4. Create statements for subjects that violate high-confidence object rules: For each rule $o_1, o_2, \ldots o_n \rightarrow o$ retrieve subjects S that violate the rule and for each $s \in S$ predict the predicate ${}^{s}p^{o}$ that connects s and the object o.

Statement creation

The step statement creation is illustrated in Algorithm 1. For each object rule $O' \rightarrow o$ with $O' = o_1, o_2, \ldots o_n$, all subjects s that occur with the antecedent of the rule but not with its consequent, ($s \in S^{O'} - S^{o}$), are retrieved in Line 2. This set contains all subjects that may be amended with new facts having the current object rule consequent o_j as their value. The choice of 90% as the high confidence threshold is arbitrary. The higher this threshold is, the fewer new statements will be generated but higher precision can be achieved. We report evaluation results on this threshold in Section 3.5.

Algorithm 1: Statement Generation Algorithm

```
   Data: objectRules /* with confidence above 90%*/
   Result: newStatements
 1 foreach objectRule ∈ objectRules do
 2     subjects ← getViolatingSubjects (objectRule);
 3     consequentObject ← objectRule.getConsequent ();
 4     foreach subject ∈ subjects do
 5         candidates ← getCandidatePredicates (consequentObject);
 6         schema ← getSchema (subject);
 7         topRating ← 0;
 8         foreach candidate ∈ candidates do
 9             currentRating←getRating (schema, candidate);
10             if currentRating > topRating then
11                 topRating ← currentRating;
12                 predicate ← candidate;
13         if topRating > $\delta$ then
14             newStatements.add (subjects, predicate, consequentObject);
15 return newStatements
```

As multiple object rules may contain the same consequent o, duplicate subject-object-pairs may be generated, which are naturally ignored. The rest of the algorithm is straightforward and starts with retrieving the candidate predicates P^{o} in Line 5 and the schema predicates ${}^{s}P$ in Line 6. The rating for each retrieved candidate predicate is computed in Line 9.

Given the set of schema elements ${}^{s}P$ and a candidate predicate $p \in P^{o}$, the confidence entries of the rule matrix are used to generate an overall rating for the specific candidate predicate p. The overall rating r_p for a candidate p is again computed by $r_p = aConf({}^{s}P, p)$, the aggregated confidence of all rules with $Q \subseteq {}^{s}P$ as antecedent and p as consequent. If the rating of the current candidate predicate is higher than the current top rating it is stored as the new top candidate. After the candidate loop, the candidate with the highest rating is returned. Only if there is a predicate with a rating above a given threshold δ, e.g., $\delta = 0$ ensures that there is any rating at all, the new fact *spo* consisting of the current subject s, the top rated predicate p, and current object rule consequent o is added to the set of new facts in Line 14. Note, the number of new facts depends on the number of existent high-confidence rules and their corresponding set of violating subjects $S^{O'} - S^{o}$.

Our approach for amending new facts can be seen as a complement approach to traditional reasoning approaches, as we do not use ontology logics but simple basket analysis. The benefit of a mining approach is that outliers and individual faulty facts do not affect the overall performance as long as the occurrence of a specific faulty fact is not statistically relevant. As a proof-of-concept we applied our mining concept to multiple datasets including the popular DBpedia data set [Bizer et al., 2009b] and provide a thorough evaluation.

3.5 Experiments and Evaluation

To evaluate the accuracy and quality of our suggestion and auto-completion approaches we performed multiple experiments to identify weaknesses and strength of our mining configurations approach. In particular we compared our approach to the most recent published system AMIE [Galàrraga et al., 2013].

In the following, we first present our experimental setup. Second, we present experiments on rule-based suggestion to evaluate our claim that predicate suggestion is more promising than object suggestion. Then, we adapt the scenario to compare the quality and efficiency of our approach with the state of the art system AMIE [Galàrraga et al., 2013], using the implementation provided by the authors[1]. The results show that our system is competitive to AMIE and achieves higher precision. Finally, we analyze different aspects of our approach and present experiments on large datasets and illustrate the different prediction capabilities of the used mining configurations and analyze the suggestion quality.

3.5.1 Experimental Setup

To present our experimental setup we first name and describe the datasets. Then we describe evaluation scenarios and corresponding quality metrics and finally we describe the experimental environment.

[1]`http://www.mpi-inf.mpg.de/departments/ontologies/projects/amie/downloads.html`

Datasets

General knowledge bases, such as DBpedia [Bizer et al., 2009b] and YAGO2 [Hoffart et al., 2013; Suchanek et al., 2013], constitute the most appropriate datasets for our use case. To identify continuous changes over time we also considered different release versions of each dataset.

Table 4.6 lists the different datasets with corresponding cardinalities that denote the number of distinct triples, subjects, predicates, and objects in that dataset. The subjects in each dataset correspond to one or more of the 250 existing types and so we are able to perform experiments not only over all entities, but also more fine-grained on entities of a certain type, such as `Person`, `Place`, or `Work`[1], resembling data of different domains. It is important to note that some types are subclasses of others. The last three datasets in Table 4.6 are cleaned knowledge bases provided by the authors of AMIE [Galàrraga et al., 2013][2]. In particular the authors removed all facts that include literal values, such as strings and numbers. We use those datasets to compare our approach to AMIE.

Table 3.1: Experimental data with distinct cardinalities

dataset	Triples	Subjects	Predicates	Objects
DBpedia 3.6	13,794,426	1,638,746	1,100	3,980,642
DBpedia 3.7	17,518,364	1,827,474	1,296	4,595,303
DBpedia 3.8	20,514,715	2,342,853	1,313	5,172,511
DBpedia 2.0	7,034,868	1,376,877	10,321	1,778,459
YAGO2	948,044	470,485	36	400,343
YAGO2s	4,125,966	1,653,882	37	606,789

Evaluation Strategy

Evaluating the quality of a prediction system under the open world assumption is difficult as no knowledge base can be seen as complete and therefore a global ground truth is missing. To verify a predicted fact would be the task of a domain expert. Our idea to evaluate fact-generating approaches is based on the insight that knowledge bases, such as DBpedia or YAGO, develop over time. Thus, we run our approach on an older version of a dataset and evaluate the generated facts by checking how many of them can be found in a more up-to-date version of the dataset. This strategy was also applied by the authors of AMIE. The ratio of the hits in the newer dataset is an indicator for the quality of a fact prediction system.

[1]Work includes "works of art", "musical work", etc.

[2]`http://www.mpi-inf.mpg.de/departments/ontologies/projects/amie/results.html`

The more facts can be found in a newer dataset, the more useful the predictions have been, as they have been added to the dataset over time by a human user. Nevertheless, in order to make the experimental results more convincing, we also applied manual inspection of the data to identify the correctness of our approach on small feasable samples.

Finally, to evaluate single mining configurations, as applied to schema or value suggestion, we also adopted the leave-one-out method evaluation approach for top-N recommendations as applied by Deshpande and Karypis [Deshpande and Karypis, 2004]. The quality measures we apply are *success rate* at x (SR@x) and the *mean reciprocal rank at x (MRR@x)*. SR@x denotes the average hit rate for finding the correct entry in a list of x elements over n experiments:

$$\textit{success rate (SR)} = \frac{\textit{Number of success}}{n}$$

The mean reciprocal rank denotes the average reciprocal hit rank of the correct element in a sorted list of size x:

$$\textit{mean reciprocal rank (MRR)} = \frac{\sum_{i=1}^{n} \frac{1}{\textit{ranking position}_i}}{n}$$

We further refer to the quality measures *precision* and *recall*, when applicable [Davis and Goadrich, 2006].

Experimental Environment

All algorithms have been implemented in Java 7. Our approach is on top of a relational DBMS where triples are organized in a relational schema *Triples(S,P,O)*. The database is only queried in the beginning of a mining configuration task to create the transactional database on the fly and to insert sorted transactions into the FP-Tree (see Section 2.1). All experiments on efficiency that are reported in this section have been performed on a Dell PowerEdge R620 machine with the following attributes:

- Operating System: CentOS 6.4
- CPU: 2 x Intel E5-2650 (2.00 GHz, Octa-Core)
- RAM: dedicated 50 GB from 128 GB DDR3-1600

3.5.2 Suggestion quality

We evaluated the accuracy of the suggestions by trying to re-suggest existing predicates or objects. We performed our evaluation for both object and predicate suggestion on the complete DBpedia 3.6 dataset, as well as on the eight types with the highest number of entities. For each dataset we randomly removed 10,000 predicates or objects respectively from 10,000 different subjects. Then we performed the suggestion algorithm on the specific dataset and checked the top-5 and top-10 recommendations for each of the 10,000 subjects. We configured our rule mining algorithm with 0.1% support and 50% confidence.

Predicate suggestion

Table 3.2a shows the results for predicate suggestion. All top-10 recommendations have a SR $> 50\%$, which means that on average in more than 50% of the test cases the removed predicate was among the top-10 results. The best success rate results are achieved on entities of type `Animal` and `Species`, where we find quite homogeneous schemata. The schemata here correspond to the ontological classifications of animals, such as `division`, `class`, and `kingdom`. The best MRR value also corresponds to the `Animal` dataset, where on average the result is at the third position. Note, these predicates are not ontologically defined designators, such as `rdfs:type` or `rdfs:subClassOf`.

The poorest result was achieved for the `Work` dataset: the correct recommendations were on average at the eighth position. As the results conform to other recommendation scenarios, such as the experiments given in [Deshpande and Karypis, 2004], we can claim that association rule mining is a reasonable strategy for predicate suggestion.

Note that even though for some scenarios the removed predicate is not in the list, it does not mean that all the proposed predicates are completely irrelevant for the specific subject. Nevertheless, by having the removed predicates among the proposed results, we can claim that a rule-based approach very often suggests the most relevant predicate for a subject. Furthermore it is possible to apply learning strategies as done by Cafarella et al. for schema auto-suggestion [Cafarella et al., 2008] on top of the statistical analysis to fit the model better for the dataset. So, there is still room for optimizing the global rule-based approach on different RDF datasets.

Object suggestion

In Section 3.3 we already hypothesized that rule-based recommendations of objects have poor quality due to the heterogeneity of objects and their large number. Indeed, the results presented in Table 3.2b confirm our statement. The best results have been achieved for the `Animal` and `Species` datasets, however the SR values at 5 and 10 are considerably lower than the values of the predicate suggestion in Table 3.2a. We can further observe that the success rates at 5 and 10

Table 3.2: Evaluations for 10,000 predicate/object suggestions per dataset

(a) Predicate suggestions

Type	SR@5	SR@10	MRR@10
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
Organisation	0.388	0.698	0.27172
Animal	0.636	0.958	0.30348
Album	0.445	0.922	0.24442
Film	0.326	0.868	0.16145

(b) Object suggestions

Type	SR@5	SR@10	MRR@10
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
Organisation	0.043	0.043	0.03543
Animal	0.451	0.551	0.21915
Album	0.005	0.005	0.00477
Film	0.034	0.035	0.02898

are always nearly identical. That means on the one hand that a few number of suggestions are very good and the hits are within the first 5 suggestion positions. In most cases however, the top 10 suggestions do not include any hit at all.

The experiments support our claim that using association rules for a suggestion scenario is much more promising on predicates than on objects. However, our approach for generating new statements that is based on high-confidence object rules is still a reasonable application of the mining Configuration 2 (Table 2.1).

3.5.3 Comparing to AMIE

Our system as well as AMIE generate new facts based on evidence in knowledge bases. While we combine two mining configurations on statement level, AMIE mines horn rules between parameterized relations. We compared both systems with regard to prediction quality as well as resource consumption.

Prediction quality

As Wikipedia infoboxes evolve over time through users creating knowledge on Wikipedia, the idea is to automatically evaluate what percentage of generated triples is included in a more up to date dataset. We used the same datasets and evaluation scenario as AMIE for a fair comparison [Galàrraga et al., 2013]. That means we ran both approaches on YAGO2 and DBpedia 2.0 and compared the predictions to YAGO2s and DBpedia 3.8, respectively. According to the original experiments reported by the developers, AMIE generates up to 74K hits in the YAGO2s dataset and 122K hits in DBpedia 3.8. However, the ratio of hits to the number of total predictions is below 1‰, as no confidence threshold was defined.

To make a fair comparison we chose the best rules generated by AMIE, that contribute the same number of predictions as our approach. To this end, we sort the horn rules generated by AMIE by their PCA (partial completeness assumption) confidence as proposed by the authors and iterate the list in descending order. We configured our approach with 90% confidence threshold for object rules and 0.1% support for both object as well as predicate rules.

The results in Table 3.3 show that our approach leads to a higher precision with regard to the test datasets. In particular, we achieve on the DBpedia dataset about 2.5 times higher precision having nearly the same amount of generated facts. On the YAGO2 dataset we report the precision for all produced rules by AMIE as the best rule already produced ten times more predictions than our approach. That specific rule generated for each relation *isMarriedTo(a,b)* generated the missing symmetric relation *isMarriedTo(b,a)* resulting in 4,424 hits in DBpedia 3.8. Comparing the two DBpedia experiments, there is a significant difference in terms of precision. One reason for the high precision in that case is because, the vocabulary of DBpedia evolves over time. That means that some properties that existed in older versions have been deprecated or renamed during the DBpedia evolution. Thus, generated facts with older vocabulary terms are less likely to be found in the latest dataset release than generated facts with more up-to-date vocabulary terms.

Table 3.3: Comparison to AMIE

Dataset	Test Dataset	Approach	Facts	Hits	Precision
DBpedia 2.0	DBpedia 3.8	AMIE (63 rules)	2,359	55	2.3%
		mining conf.	2,335	146	**6.2%**
YAGO2	YAGO2s	AMIE	1.658m	8.1k	0.5%
		mining conf.	2,086	52	**2.5%**
DBpedia 3.6	DBpedia 3.8	AMIE	-	-	-
		mining conf.	26,660	8.2k	**30.0%**

Of course the results only confirm that some facts are true, but cannot confirm that any of the generated facts are false unless checked by a human expert. Due to memory consumption restrictions (50GB) we could not evaluate AMIE on the original DBpedia 3.6 dataset.

Resource consumption

AMIE is a multithreaded approach where the knowledge base is kept and indexed in main memory to compute support and PCA confidence values in appropriate time. The drawback is clearly the high memory consumption that requires up to 22 GB to discover rules on the DBpedia 2.0 dataset and 3.4 GB for the YAGO2 dataset. Our mining configuration system needs only two FP-Trees in memory (one for discovering predicate rules and one for object rules), resulting in less than 600 MB when running on DBpedia 2.0 and about 200MB for YAGO2. We perform both steps, mining predicates and objects, consecutively, which could just as well be done in parallel to improve the runtime. The runtime of both approaches, AMIE as well as our mining configurations, on these datasets is under 1 minute. Our approach consumed less than 20 GB memory for the DBpedia 3.6 dataset and the experiments of the next subsection.

In general, we conclude that both approaches are valid strategies to amend a knowledge base with new facts. While AMIE generates new facts based on closed rules considering entire fact patterns as rule atoms, our approach is more granular in considering predicate correlations and object correlations independently. As AMIE's performance is based on a customized in-memory database, it consumes much more main memory than our configuration based approach. In the following, we further analyze the quality of our approach on larger datasets and on domain specific data samples.

3.5.4 Amendment quality on large datasets

Depending on the support and confidence thresholds and the underlying DBpedia fragment, our algorithm generates up to tens of thousands of new triples. To further analyze the capabalities of rule-based triple amendment we performed more experiments on the DBpedia 3.6 dataset. To identify strengths and weaknesses of the approach we also performed experiments on subsets of that dataset. Table 3.4 shows the number of generated facts and their inclusion ratio in the DBpedia 3.7 dataset. The algorithm parameters have been 0.1% support for object and predicate rules and 50% and 90% confidence for predicate rules and object rules, respectively. Note, the listed types are not disjoint. For example, the new facts generated on the basis of `Animal` instances might also include some of those facts that have been generated on the basis of the `Species` instances.

The high precision of the results for animals is caused by the fact that most of the newly added statements are animal classification statements that have been missing in the older version because of the lack of thoroughness during data creation. Note, these classification statements do not correspond to the ontology class desginators `rdfs:type`. Those statements have been excluded to identify more interesting new facts.

Table 3.4: Generated statements on DBpedia v3.6 and their inclusion in v3.7

Type	Facts	Included	Version-Precision
Thing	26,646	8,225	31.2%
Person	1,521	278	18.3%
Actor	50	6	10.5%
AdministrativeRegion	66	7	10.6%
Airport	363	78	21.5%
Album	43	25	58.1%
Animal	17,024	8,753	51.4%
Artist	437	225	51.5%
Athlete	890	84	9.4%
Eukaryote	39,115	15,999	40.9%
Film	280	34	12.1%
MusicalWork	99	34	34.3%
Organisation	1,465	265	18.1%
Place	13,090	1,451	11.1%
PopulatedPlace	13,365	1,568	11.7%
Settlement	13,274	1,351	10.2%
SoccerPlayer	1,025	59	5.8%
Species	39,209	15,987	40.8%
TelevisionEpisode	42	3	7.1%
Village	1,333	17	1.3%
Work	466	66	14.2%

The 3.7 dataset contains 3,723,938 more triples than the DBpedia 3.6 dataset, of which 2,410,080 triples are facts that are recombinations of subjects, predicates, and objects from the 3.6 dataset and therefore in principle predictable by our algorithm. Performing our algorithm with the above parameters on the complete 3.6 dataset, we are able to generate 26,646 new facts out of which 8,325 are contained in the 3.7 dataset: We were able to guess 0.4% of all new facts in the new DBpedia version that are added by multiple real Wikipedia users or editors and can be generated by recombining existing statement parts. The ratio is even higher if we perform the amendment algorithm on datasets that correspond to

more special types (see Results for `Animal` in Table 3.4). While having 31.2% precision for *minconf* = 90%, experiments on the complete dataset (all entities of type `Thing`) with thresholds of 95% and 85% resulted into 44.3% precision having 5,866 new facts and 27.4% precision having 39,589 new facts, respectively. These results confirm our assumption that the higher this threshold is set the more precision can be achieved but the fewer facts may be generated. Those facts that were not included in DBpedia 3.7 are not necessarily wrong facts. We additionally evaluated manually a random set of 50 not-included facts and achieved 72% precision.

Table 3.5 illustrates the most frequent predicates among the correctly generated facts of the different domains, `Species`, `Person`, and `Album`. While some of the predicates, such as `class` or `kingdom`, could be deduced via specific taxonomies, our approach also generates facts that cannot obviously be deduced from existing ontologies, e.g., facts with the predicates `battle` or `birthPlace`.

Table 3.5: Predicates that appear in generated correct facts

Person		Work		Species	
predicate	#	predicate	#	predicate	#
background	204	language	58	kingdom	4,513
birthPlace	34	computingPlatform	4	phylum	2,560
battle	24	country	4	class	2,222
statisticLabel	13	computingInput	1	order	126
award	1			family	26
				conservationStatusSystem	1

We performed the previous set of experiments for a second time and checked only whether the subject-object pair without the assigned predicate was included in the new dataset. Interestingly the precision only marginally improved (e.g., 8,296 pairs included for `Thing`), implying that the accuracy of our experiment is bound to the truth of high-confidence rules. Therefore, we further analyze the impact of each configuration on the quality of our approach.

How true is a high-confidence rule?

Our intuition about high-confidence rules is that those subjects that violate these rules are actually not intended to violate them. In other words, we assume that the number of those subjects that deliberately "violate" the rules is relatively low. We evaluated the quality of high-confidence rules $o_i \rightarrow o_j$ by manually verifying the relation of its consequent o_j to the violating subjects and identifying the ratio between real exceptions in the data and unintentionally absent values. Because of the large number of rules and violating subjects, we performed the manual verification on only 50 randomly selected violating subjects per dataset.

Note that these samples are different from the random sample used in the previous section. To generate the sample, we iteratively selected random high-confidence rules. For each such rule, we randomly selected up to five violating subjects until we reached the limit of 50. We then manually checked whether the subjects are related to the rule's consequence object by reviewing the corresponding Wikipedia pages of the alleged violating subjects and checked whether the consequence of the high-confidence rule is related with the subject or not. As the underlying DBpedia data is less up to date than the Wikipedia pages, some of the up-to-date infoboxes included the object values that were missing in the DBpedia 3.6 dataset, which made it easier to review the pages manually.

Table 3.6 shows the results from the manual evaluation for 50 randomly chosen subjects from four datasets. Each dataset corresponds to entities of the given type from DBpedia 3.6. The generated object rules for this experiment hold 0.1% support and 90% confidence. We observe that the assumption holds for most objects rules in the domains `Person` and `Place`, such as *American Civil War* $\rightarrow$ *United States* for `Person` instances and *Vosges* $\rightarrow$ *Lorraine Region* for `Place` instances. High-confidence rules from movie data however are mostly the result of true exceptions: the rule *Felix Adler* $\rightarrow$ *Moe Howard* with 93% confidence. Although the screenwriter Adler is strongly connected to the actor Moe Howard (The Three Stooges), he also had movie projects without him. However, in movie data there are also interesting positive examples: the rule *Lon Chaney, Sr.* $\rightarrow$ *Silent Film* with 93% confidence is a rule where the violating movies are silent movies and can be updated with the object *Silent Film*, because *Lon Chaney Sr.* acted only in silent movies.

Table 3.6: Percentage of true violations of a high-confidence rule

Type	Thing	Place	Person	Film
True Positives	37	41	42	22
Percentage	74%	82%	84%	44%

Note that the amendment algorithm creates a new fact with the missing object only if there is a predicate that matches the violating subject and the object rule consequence. In the following, we evaluated the completion with predicates.

Completion with Predicates

Given a subject s, its schema ${}^{s}P$, and a related object ${}^{s}o$, the aim of predicate completion is to select the most appropriate predicate ${}^{s}p^{o}$ out of all predicates

P^o that have o in their range. We evaluated this step by applying the leave-one-out strategy: For each high-confidence object rule $o_i \rightarrow o_j$ we considered all subjects s^{o_j} that do not violate this rule and removed the connecting predicate ${}^{s}p^{o_j}$ between the subject s and the consequence object o_j and tried to predict ${}^{s}p^{o_j}$ based on the predicate matrix and the predicate candidates P^o.

Table 3.7 illustrates the results for experiments on the complete dataset (`type:Thing`) as well as the eight types with the most instances. In comparison to the suggestion evaluation, we see that the choice of the correct predicate is very accurate when knowing also the object of the statement. We achieve lower precision on `Person` because many object rules there refer to locations, such as *Buenos Aires* $\rightarrow$ *Argentina*, and the predicate selection confuses predicates, such as `nationality`, `deathPlace`, and `birthPlace`. The most general and frequent predicates often receive a higher ranking than the more special predicates. But even though the removed predicate is confused for these examples, the proposed predicate for the subject-object pair might still be a valid fact.

We mentioned earlier that for those subject-object pairs, where the existing schema of the subject ${}^{s}P$ and the candidate predicates P^{o_j} are not related to each other by any predicate rule, the algorithm ignores those subject-object pairs. The column with the number of missing values represents the number of subject-object pairs, for which the algorithm does not select any predicate. For those pairs, the existing schema of the subject ${}^{s}P$ and the candidate predicates P^{o_j} are not related to each other. Because only few triples are concerned, the precision is always at least as high as the recall. Only for the `Film` dataset where the algorithm decided to suggest no predicate for four subject-object-pairs, the precision is slightly higher than the recall. One could assume that by increasing the minimum threshold for the selection decision (see Alg. 1, line 13) incorrect selections can be avoided by being marked as undecidable. However, experiments showed that increasing the threshold results in more undecidable selections and in fewer correctly selected predicates.

Finally, we evaluated the predicate selection based on randomly removed predicates. We wanted to examine whether the quality of the predicate selection depends on the choice of the objects and whether the fact that they are connected with consequences of high-confidence object rules influences the quality. Furthermore, this scenario corresponds to the use-case where a user already knows an object value that should be added to a subject and looks for the correct predicate.

Table 3.8 illustrates the results for predicting randomly removed predicates. The results show that the precision and recall values decrease only marginally for most datasets but also marginally improve on some datasets, such as `Person`, `Album`, and `Film`. Overall the results imply that predicate selection does not depend on the choice of objects as there is no significant difference to the results in Table 3.7.

Table 3.7: Results for predicting removed predicates based on object rules

Type	Rules	Removed	Correct	Missing	Incorrect	Precision	Recall
Thing	189	1,019,785	919,815	1	99,969	90.2%	90.2%
Place	169	246,731	246,704	0	27	99.9%	99.9%
Person	37	30,120	20440	0	9,680	67.9%	67.9%
Work	10	5,725	5,669	0	56	99.0%	99.0%
Species	1,128	1,337,734	1,212,080	9	125645	90.6%	90.6%
Org.	84	24,535	24,258	0	277	98.9%	98.9%
Animal	981	952,340	951,964	8	368	99.9%	99.9%
Album	6	782	683	0	99	87.3%	87.3%
Film	50	5,618	5,086	4	528	90.6%	90.5%

Table 3.8: Results for predicting 20,000 random predicates for each type

Type	Predictions	Correct	Missing	Incorrect	Precision	Recall
Thing	18,731	16,855	1,269	1,876	89.98%	84.28%
Place	19,775	18,359	225	1,416	92.84%	91.80%
Person	19,936	15,419	64	4,517	77.34%	77.10%
Work	19,865	17,291	135	2,574	87.04%	86.55%
Species	19,986	17,922	14	2,064	89.67%	89.61%
Organization	19,820	16,115	180	3,705	81.31%	80.58%
Animal	19,975	19,968	25	7	99.97%	99.84%
Album	19,861	19,121	239	640	96.27%	95.61%
Film	19,842	18,606	158	1,236	93.77%	93.03%

Completion with Objects

We also evaluated the completion step for randomly removed objects. As Table 3.9 illustrates, precision and recall for object selection is far below the values for predicate selection. Predictions for `Thing`, `Work`, and `Album` resulted in zero precision. This is caused by the quantitative relations of predicates and objects. Indeed, for each subject predicate pair there are either many objects that are potential candidates for the selection or no candidates at all, because the fact generation algorithm (Algorithm 1) could not retrieve a rating above zero. Note, many object values are literals, which have very low frequencies in the dataset. For example, in our experiments on the `Person` dataset, the average number of candidate predicates per subject-object pair was 4, while the average number of candidate objects per subject-predicate pair was 842.

In general, we conclude that mining configurations is a reasonable approach to enrich RDF data. Comparing to state-of-the-art systems it consumes fewer

Table 3.9: Results for predicting 20,000 randomly removed objects for each type

Type	Predictions	Correct	Missing	Incorrect	Precision	Recall
Thing	0	0	20,000	0	0%	0%
Place	5,761	4	14,239	575	0.07%	0.02%
Person	2,400	1,408	17,600	992	58.6%	7.04%
Work	800	0	19,200	80	0%	0%
Species	3,200	1,722	16,800	147	53.81%	8.61%
Organisation	1,132	84	18,868	104	7.42%	0.42%
Animal	4,000	1,837	16,000	216	45.93%	9.19%
Album	0	0	20,000	0	0%	0%
Film	400	203	19,600	197	50.75%	1.02%

resources. Furthermore, the overall precision of our approach is much higher allowing a manual verification step after generating new facts. The quality of our approach highly depends on the reliability of object rules, as the predicate selection step works pretty accurate. In a real-world scenario it is possible to drop object rules that denote weak hypotheses. The generated facts and an online demonstration tool embedding our approach can be found on our website[1].

3.6 Summary

Putting our mining configurations into use, we elaborated the two configurations with "subject" as their context and introduced two algorithms for enriching RDF data and enhancing its quality. We showed how an association rule matrix can be used for suggesting both predicates as well as object values for a user who is in the process of inserting new statements for an entity. Further, we proposed a user-driven and a data-driven approach for generating new facts without depending on external resources. The key intuition for avoiding the generation of masses of faulty facts was to constrain the system based on high-confidence object rules.

We showed that our solution differs from previous work as it is among the few approaches that consume a single source of linked data without annotated or specified logical rules to generate new knowledge. Our experimental results showed that the predicate suggestion use case as well as the data-driven auto-amendment approach lead to good results. Comparing to the most recent state-of-the-art approach AMIE, mining configurations is competitive in terms of quality and resource consumption. Furthermore, we identified special domains such as animals and plants where mining configuration achieves higher precision than more general knowledge bases.

[1]`http://www.hpi.uni-potsdam.de/naumann/projekte/mining_rdf_data.html`

Our current roadmap on enriching RDF data (Figure 3.1) resembles a situation where the to be amended subject is already given. However, considering the SPO structure in a more abstract way and without the semantic connotation of the specific statement position of the resource, our mining configuration methodology theoretically enables us to generalize the approach on enriching data so that the object or the predicate can also be picked as initial resources that can be enriched with remaining missing parts. But in practice such an approach would face several challenges. First of all, such an approach requires different intuitions than the current intuition for creating new triples. When considering the subject of an SPO triple, we implicitly talk about an entity that can be amended with new facts, which are derived from other similar entities. Applying the same intuition to amend predicates and object values is not obvious. Secondly, as already discussed in previous sections, identifying valid support and confidence thresholds for promising results in different configuration contexts requires additional effort and intuitive justification.

In the next chapter, we further elaborate on improving RDF data by means of frequency analysis and rule mining to reconcile instance data and ontological definitions.

Chapter 4

Improving Ontologies

One of the main goals of Linked Open Data (LOD) is to enable information providers to create and share linkable data across the Web. Beside the increasing number of LOD sources, there also exists a set of rules for publishing Linked Data [Heath and Bizer, 2011]. Still, consuming and integrating LOD necessitate a thorough analysis and study of the data sources, because individual data providers have different understandings or knowledge of appropriate vocabulary definitions. A recent survey of Linked Data conformance [Hogan et al., 2012] analyzed the degree of conformance of data to different existent guidelines, showing that most data providers adhere to syntactical guidelines, such as the use of stable URIs. However, regarding ontology usage and property consistency even well known datasets, such as DBpedia or the Music Ontology[1], suffer from lack of metadata conformance.

Here, reusable knowledge bases and ontologies facilitate comprehending and integrating multiple data sources. These ontologies provide metadata to define the domains and ranges of properties of resources or taxonomical relationship between these resources. In our work, we analyze the differences between the specification and the usage of such vocabularies and offer a data-driven approach to refine existing ontologies. In the previous chapter, we haven already shown how rule mining can be used to infer new facts or support fact generation in order to maintain a higher degree of consistency and completeness in data. Here, we demonstrate how rule mining can also be used to identify misusage of ontology axioms and properties and how such issues can be resolved following a statistical approach.

We observed a significant distinction between well-defined ontologies and the common understanding of LOD. While LOD in general is designed to be easily extensible so that facts about resources in one dataset can be added ad-hoc and

[1]`http://semanticweb.org/wiki/Music_Ontology`

independent of other sources, ontologies define structural information for data sources of a specific domain and are therefore less flexible. Hence, integration of LOD, ontology discovery, and matching pose major challenges for the pervasion of the Web of Data. While there are best practices for publishing Linked Open Data using established ontologies [Heath and Bizer, 2011], our analysis shows 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. Thus, custom namespace-specific properties are often added to an ontology concept when need for it arises. It is likely that some of these properties are redundant, as data providers are unaware of each other's additions. Here, one can also refer to "ontology hijacking" [Hogan et al., 2012].

Addressing such quality issues of an ontology can be considered a form of schema and range analysis. Here, a schema defines the set of properties whose domain is a specific class within the ontology. Hence, redesigning ontologies requires analysis and mining of the underlying data. We apply data mining by applying our mining configuration methodology and present an automated data-driven approach that generates suggestions to support ontology adjustment. This chapter includes insights and case studies from our research paper [Abedjan et al., 2012] and our proposal for the Billion Triple Challenge 2011 [Lorey et al.][1], which were co-written with Johannes Lorey.

The remainder of this chapter is organized as follows: In the next section, we address the existing research directions on ontology alignment and improvement. In Section 4.2, we describe and define the concrete ontology misusage patterns that our approach rectifies. In Section 4.3 illustrates our algorithm that generates the adjustment suggestions based on usage patterns. Finally, we evaluate our re-engineering approach on two large-scale datasets, namely DBpedia and the crawl for the Billion Triple Challenge (BTC) 2011 in Section 4.4.

4.1 Related Work

While ontologies have been a core concept of the Semantic Web and artificial intelligence, most research concentrates on conceptual techniques and ontology languages that enable to implement conceptual vocabularies or align multiple ontologies [Guarino, 1998; Aguirre et al., 2012]. In fact there are only few works that analyse vocabulary design and its usage.

Usage based ontology alignment was already introduced by [Stojanovic and Stojanovic, 2002], which was integrated into the tool OntoManager [Stojanovic et al., 2003]. Similarly, Haase et al. follow a collaborative filtering approach based on usage to improve ontologies [Haase et al., 2005]. In their terms, usage refers to

[1]Our proposal ws selected as one of the finalists of the challenge

the interaction of a user with a given ontology with regard to manually traversing links in the graph taxonomy or changing them for personal purposes. However our approach refers to ontology usage in terms of occurrence of properties as predicates in the dataset. Furthermore, all three approaches concentrate on taxonomy inconsistencies and none of them is tailored to RDF data. The OntoClean methodology evaluates ontologies and ontological decisions [Guarino and Welty, 2002]. However, the authors discuss ontological decisions from a conceptual view, proposing to define metaproperties that better characterize the meaning of vocabulary elements of an ontology. Our work differs from that approach, as we analyze actual ontology usage and present an algorithm to evaluate and improve an ontology by considering its usage by the community.

There have been a number of case studies about the pervasiveness and the usage of certain RDF vocabularies [Ding et al., 2005; Golbeck and Rothstein, 2008], which are related to our work. In particular, the first study analyzed which FOAF properties were frequently used and the authors consider the findings as guidance to improve the FOAF vocabulary [Ding et al., 2005]. We follow a similar intuition but present a concrete approach that suggests ontology re-engineering decisions for any given ontology.

The system presented in [Kinsella et al., 2008] provides users with an interactive visualization of the specifications and connections between different ontologies based on an RDF dataset, allowing ontology engineers to discover possible design flaws. In our work, we focus not only on just discovering the divergences between ontology definitions and their usage, but also present a data mining based approach to resolve these discrepancies.

Another stream of research acknowledges inconsistencies in RDF data with regard to ontological vocabularies, such as domain and range definitions, and proposes methods to resolve the inconsistencies or to repair the underlying ontology [Töpper et al., 2012; Knuth et al., 2012]. Knuth et al. propose a collaborative method based on a quiz game to identify data inconsistencies [Knuth et al., 2012]. The collected data is presented via a patch repository that can be used to update ones ontology. The obvious drawback of that method is that it relies on an active crowd. Töpper et al. look for inconsistencies of `rdfs:domain` and `rdfs:range` definitions with regard to the actual `rdfs:type` of corresponding subjects and objects [Töpper et al., 2012]. In order to resolve the identified inconsistencies, they treat each inconsistency occurrence as an individual case that has to be rectified by changing the corresponding axioms. Our solution rather concentrates on removing inconsistencies by relocating properties and redefining their ranges and domains. Furthermore, our approach considers statistical heuristics based on all instances of the relevant class.

Several works in the field on ontology engineering aim at establishing and enriching ontology specifications by using machine learning techniques [Buitelaar and Cimiano, 2008; Lehmann and Bühmann, 2010]. The authors of [Maedche and Staab, 2001] present a semi-automatic approach for cross-domain ontology learning. Similarly, in [Wu and Weld, 2008] machine learning methods are employed to refine the definition of the Wikipedia infobox-class ontology. In contrast to these works, our approach allows incremental re-engineering of an existing ontology based on provided instance data without any training data or third party data sources, such as WordNet.

Finally, the authors of [Völker and Niepert, 2011] present a schema induction approach based on association rules to recreate axioms of the DBpedia ontology. As their approach generates an ontology axiom for every generated rule, redundant or conflicting axioms are created, too. Our approach differs from their work in two major aspects: First, we do not want to recreate an ontology but to improve a given ontology, by proposing the removal of unnecessary axioms or inclusion of appropriate missing axioms, based on identified misusage patterns. Second, we have define constraints that allow our approach to create sound suggestions for improving an ontology definition by avoiding the generation of redundant and conflicting suggestions.

4.2 Problem Statement

Assessing the quality of an ontology is a very difficult task, because the structure of an ontology and its usability well depend on the specific dataset domain and the semantic intention. Therefore, it is difficult to define a holistic quality measure. However, in case of LOD one can address bad practices and identify specific misusage patterns. To illustrate the problem of discrepancy between ontology specification and usage, we begin with a motivational example and then introduce the related misusage patterns.

Table 4.1: RDF prefix abbreviations used in this work

Prefix	Ontology namespace
`foaf:`	`<http://xmlns.com/foaf/0.1/>`
:	`<http://dbpedia.org/ontology/>`
`rdfs:`	`<http://www.w3.org/2000/01/rdf-schema#>`
`owl:`	`<http://www.w3.org/2002/07/owl#>`
`dbp:`	`<http://dbpedia.org/property/>`
`rdf:`	`<http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
`mo:`	`<http://purl.org/ontology/mo/>`
`mb:`	`<http://musicbrainz.dataincubator.org/schema/>`
`dc:`	`<http://purl.org/dc/terms/>`

4.2.1 Property Misusage Example

For brevity and readability, we use a number of prefix abbreviations when denoting RDF resources. These abbreviations are defined in Table 4.1.

Listing 1 shows an excerpt of the `:Settlement` class definition from DBpedia 3.9. If a geo-location data provider decides to publish her data using this specification, she might be confused about how to set proper values for some of the properties. The properties `:winterTemperature` and `:summerTemperature` are intuitively applicable as predicates to all instances of `:Settlement`, whereas others seem only useful for a strict subset of instances, such as `:scottishName` and `:distanceToEdinburgh` in Lines 4 and 5. Note, in the context of class definitions we talk about properties. At instance-level, i.e., when talking about specific RDF facts, properties are used as predicates. Looking at the property `:canton`, it is also intuitively not clear why such a specific property should have Settlement in its range while a Canton semantically describes more specific locations.

On the other hand, none of the properties of the class `:Settlement` (or any of its parent classes `:PopulatedPlace`, `:Place`, and `owl:Thing`) model the latitude and longitude degrees of a settlement although these are set for many instances of `:Settlement` in DBpedia, e.g., via the `dbp:latd` and `dbp:longd` predicates, respectively.

Overall, for a more intuitive vocabulary definition some of the properties (e.g., `:scottishName`) could be removed, delegated to a more suitable subclass of `:Settlement` such as `:ScottishSettlement` if available, or marked as optional if supported by the ontology language. Additionally, some of the predicates that are already set for a large number of instances of `:Settlement` (e.g., `dbp:latd`) may be considered to be included in the class definition.

```
1 :Settlement           rdfs:subClassOf :PopulatedPlace .
2 :winterTemperature    rdfs:domain     :Settlement .
3 :summerTemperature    rdfs:domain     :Settlement .
4 :scottishName         rdfs:domain     :Settlement .
5 :distanceToEdinburgh  rdfs:domain     :Settlement .
6 :canton               rdfs:range      :Settlement .
```

Listing 1: Excerpt of the `:Settlement` specification.

In general, the denoted discrepancies may cause confusion which ontology or classes therein to adopt when publishing RDF data. Using vocabularies that are not intended for certain resources or unwarranted extensions to existing ontologies limits the machine readability and thus impedes the benefits of Linked Data. The goal of our work is to identify recurring misuse of ontology definitions and to help overcome these problems by offering re-engineering suggestions.

4.2.2 Misconception Patterns

Divergences between ontology specification and instance data may occur in multiple scenarios: On the one hand, the ontology might have been devised independently and before data that uses it was published, e.g., in the case of the "Friend of a Friend"[1] ontology (FOAF). On the other hand, the ontology might have been tailored for existing data, e.g., in the case of the DBpedia project. Last but not least, crowd source based ontology definitions as it is encouraged by the DBpedia community[2] may also result in inconsistent property or class definitions. We identify two typical cases where the specification of an ontology differs from usage patterns: *overspecification* and *underspecification*.

Both overspecification and underspecification may stem from data providers being unaware of the specifics of the ontology they employ for the data they publish. Thus, they may neglect certain properties (although suitable) or introduce new ones (even though these may be semantically equivalent to the existing ones). However, in our work we focus on ontology engineers by offering usage information as well as re-engineering suggestions to them. Furthermore, we adapt the concept of over- and underspecification with regard to the ranges of properties, where range definitions may be to specific or too general.

Overspecification

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`.

Over time, RDF ontologies may grow by introducing new class and property definitions. Especially cross-domain ontologies, such as DBpedia and Yago, have evolved extensively since their first specification. However, revising existing class definitions is sometimes neglected during this evolutionary process or its actual usage is not being considererd by the ontology engineer. This leads to several problems:

- Data providers cannot set proper values for the defined properties, e.g., a `:scottishName` for a non-Scottish `:Settlement`.
- There are multiple semantically equivalent properties defined for a class, e.g., `:occupation` and `:profession` for `:Person`.

In most cases for which we identified overspecification, one solution is to remove properties from the class concerned. However, sometimes these properties

[1] `http://xmlns.com/foaf/spec/`
[2] http://wiki.dbpedia.org/Support

are still valid for certain subclasses of this class. In particular, subclasses might have been added and (inherited) properties are now used only for instances of these, e.g., `:philosophicalSchool` used exclusively with `:Philosopher`, but defined for parent class `:Person`. Thus, before removing a property from the ontology it is essential to verify whether it is used in instances of any subclass. If so, the property needs to be *pushed down* to the appropriate subclass.

Underspecification

We deem a class to be underspecified, when in real-world data certain properties are used frequently even though they are not specified by the vocabulary, e.g., `dbp:latd` for `:Settlement`.

The flexible RDF data model allows the ad-hoc assignment of predicates to instances of a certain class. However, if this predicate is suitable for a large number of instances, it might prove beneficial to add it as a property to the corresponding class definition. Doing so aids data integration and ensures that other data providers are aware of this property. The reasons for underspecification include:

- The class definition lacks certain properties that are commonplace in instance data, e.g., `dbp:latd` for `:Settlement` or `:genre` for `:Band`.
- 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 is also used for instances of other subclasses of `:EducationalInstitution` such as `:College`.
- Data providers extend the ontology independently of one another, e.g., 150 properties with various namespaces have domain `foaf:Person` in the BTC 2011 crawl[1] whereas FOAF only defines 16 such properties.

To overcome underspecification, properties may be added to a class definition. Clearly, if each class of an ontology is checked separately, it may happen that properties are suggested for a class as well as some of its subclasses. In our approach, we ensure that our suggestions do not create unnecessary redundancy. If both a class and several of its subclasses are underspecified with respect to a certain property, this property is not added to these subclasses, but *pushed up* to the common parent class instead. Moreover, to avoid that properties of a different ontology are added to a class because of simple wrong class assignment of entities, it is important not only to consider the frequencies of the predicates but also their dependency on already defined properties.

[1]`http://km.aifb.kit.edu/projects/btc-2011/`

It is important to note that a class can simultaneously be overspecified and underspecified (with regard to different properties). Therefore, an approach should combine the analysis for under- and overspecification so that the results are consistent and without loss of information.

Over-/Underspecification of Range Definitions

So far, we discussed over- and underspecified classes with regard to the set of properties that describe instances of that class, i.e., instance of that class occur in triples in the role of subjects while the properties are used as predicates. Another equivalent view to the problem is to refer to the domain specification of a property. In RDFS, properties are instances of the class `rdf:Property` and have RDFS properties that specify the domain and range of a given property, namely `rdfs:domain` and `rdfs:range`. Thus, whenever we mention the property P of some class C, there is a triple stating:

P `rdfs:domain` C.

Similarily in RDFS the range of a property P is defined by a triple of the form:

P `rdfs:range` C.

In fact, it is also possible to identify ontology misusage with regard to the range of properties. The range of a property might have been over-specified by a class, when the real range of instances that occur as objects in triples with the specific property as predicate are not instances of the specified class. Or the range might be underspecified by a general class definition. One could argue that the latter is not a significant problem, as long as it is ensured that object values with the certain predicate are always captured within the class that is specified in the `rdfs:range` definition. However, especially with regard to reasoning, a too general range class causes runtime and quality problems.

In the next Section, we present a general approach to identify the above presented misconceptions with regard to `rdfs:domain` definitions and demonstrate how it can also be easily adapted to identify misusage patterns with regard to `rdfs:range` definitions. In both cases our approach makes useful suggestions to rectify the misusage patterns by adapting the ontology definitions.

4.3 Data-driven Reconciliation of Ontology and Data

We present an approach that can be used to improve class definitions, i.e., domain definition of properties, as well as range definitions of a property. As the methodology for the identification of both ontology misusage scenarios is analogous, we explain our approach with regard to the problem of over-specification and under-specification of classes. At the end of this section, we point out differences when applying our system to suggest re-engineering options for range definitions of properties.

4.3.1 Algorithm Overview

To determine data-driven ontology re-engineering options, one could align specific classes and associated instance data to identify predicates to be removed from or added to the class definition, respectively. However, this would have two limitations: (1) some properties might be removed instead of being allocated to a more suitable subclass, and (2) some properties might be added to a class and its subclasses independently without regarding inheritance relationships. Therefore, we propose a holistic approach that processes all classes of a given ontology by also considering class hierarchies.

Our approach to generate suggestions for predicate removal and inclusions consists of the following steps:

1. Identify typed entities in the data (declaring `rdf:type`) and retrieve relevant ontological information, such as subclass dependencies.
2. Generate removal suggestions by detecting rarely used properties.
3. Generate inclusion suggestions by mining predicates

Depending on the dataset, the first task is more or less straightforward: For instance, in the case of DBpedia, explicit instance mapping information as well as a well-defined ontology are available and easy to consume. Having an unknown data set, such as the BTC dataset on the other hand, type information has to be extracted and corresponding ontology information has to be retrieved online. The latter two steps describe our principal approach for generating re-engineering suggestions. In the following, we present the intuition of our approach and the procedure of making ontology adjustment proposals.

Given a dataset with typed instances and a corresponding ontology, we apply frequency and association rule analysis to identify over- and underspecification. Here, we apply Configuration 1 of our mining configurations for instances of each existing ontology type, which means that we mine predicates in the context of subjects. We are interested in frequent patterns as well as association rules that can be derived applying Configuration 1. Frequency analysis enables us to identify the usage of certain properties, while the association rule analysis ensures to identify useful inclusion suggestions. Next, we describe the two consecutive steps for the ontology adjustment: Generating property removal and property inclusion suggestions.

4.3.2 Property Removal Suggestions

Identifying cases of overspecification in a class definition is straightforward: Given a minimum support threshold of s, each property that does not hold s (the property occurs for less than $s\%$ of all instances of a class) in the given data constitutes an overspecification of the current class and should therefore be suggested for removal from 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. The set of all properties that do not meet the minimum support are marked for removal from the specific class definition.

The choice for minimum support is the key parameter for our approach. The higher it is chosen, the more properties might be suggested for removal. In real-world data the minimum support should be set very cautiously, because low coverage of a property among instances of a class does not necessarily mean that the property is wrongly assigned. Other reasons should be considered, too. For example, actual facts might still be missing or the data might contain exceptions that are difficult to fit into an exact ontology. Thus, when we consider over-specification, we refer to a significantly large amount of missing property values, i.e., over 99% of entities of a certain type have not set a certain property. Or in other words, the property does not have a support above 1%. For such a scenario, the property should usually be attached to a subclass instead.

4.3.3 Property Inclusion Suggestions

Having marked removal suggestions, we now determine 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. In order to propose relevant properties to the correct classes we have formulated three important heuristics, which are explained in the following.

To avoid suggestions that are caused by wrong type assignments of third party data, we have to ensure that only those properties are added to a class that are highly correlated with already (validly) defined properties of this class. For this purpose the following Heuristic 1 incorporates association rules between predicates:

Heuristic 1 *If for two predicates p' and p there is a rule $p' \rightarrow p$ with minimum support s and minimum confidence c, and domain(p') $= \mathcal{C}$ and domain(p) $\neq \mathcal{C}$ and domain(p) $\neq \mathcal{P}_i$ for all of $\mathcal{C}$'s strict parent classes $\mathcal{P}_1, \mathcal{P}_2, \ldots$, predicate p should be proposed as a property for $\mathcal{C}$.*

Using the domain restrictions in Heuristic 1 we avoid suggesting frequent predicates that are used because of wrong type assignment of the associated instances, i.e., instances where those predicates are not included in the class definition. Furthermore, the suggestions are more accurate when having multiple options within a subclass hierarchy branch.

It may happen that under Heuristic 1 a predicate is proposed for a class $\mathcal{C}$ as well as for one or more of the subclasses of $\mathcal{C}$. For example, the predicate `:anthem` may hold enough support among the instances of type `:Place` and may also be associated with properties defined for `:Place`. However, it might be the case that most or all those instances of type `:Place` with `:anthem` are in fact instances of the more specific type `:Country` (which is a subclass of `:Place`). Then the more specific class is more suitable to constitute the domain of that property. Therefore, we define Heuristic 2 based on which the algorithm proposes the property to be added to the more specific class. On the other hand, when a predicate such as `:populationDensity` is proposed for multiple subclasses of `:PopulatedPlace`, such as `:Country`, `:City`, and `:Continent`, as well as for `:PopulatedPlace` itself on behalf of Heuristic 1, it is more appropriate to propose the predicate for the more general class `:PopulatedPlace`. Heuristic 3 defines what our algorithm would decide in that case.

Heuristic 2 *If Heuristic 1 holds for predicate p and class $\mathcal{C}$ as well as for p and exactly one of $\mathcal{C}$'s strict subclasses $\mathcal{S}_i$, property p should be proposed for the subclass $\mathcal{S}_i$ instead of $\mathcal{C}$.*

Heuristic 3 *If Heuristic 1 holds for predicate p and class $\mathcal{C}$ as well as for p and more than one of $\mathcal{C}$'s strict subclasses $\mathcal{S}_1, \mathcal{S}_2, \ldots$, where more than one of these subclasses are not subclasses of one other, p should be proposed only for $\mathcal{C}$.*

Algorithm 2 illustrates the workflow for generating property inclusion suggestions after the identification of removal candidates. The input of the algorithm includes the complete set of instance triples (i.e., triples about resources for which the type is known) denoted as *triples* as well as the set of classes *classes* of the ontology to be adjusted. For each class c it is indicated whether properties have been defined specifically for the class or inherited from parent classes and whether the properties are removal candidates. The result of the algorithm are enriched class definitions containing inclusion and removal suggestions for each class.

It may happen that one predicate is an inclusion candidate for a class as well as for some of its subclasses. To decide for which class specifically such a property should be proposed, it is necessary that all subclasses have been analyzed before the parent class. This is ensured by the reverse topological sorting of *classes* in line 1. Based on the topological sorting, the algorithm traverses the

Algorithm 2: Property Inclusion Suggestion Algorithm

```
Data: classes : class definitions (including original and cleaned schema)
Data: triples : instance triples for all available classes
Data: minSupp, minConf : minimum support and confidence values
Result: classes : class definitions enriched with property suggestions
1  classes.topologicalSortAscending ();
2  foreach $c \in classes$ do
3      schema ← c.cleanProperties ();
4      inheritedSchema ← c.cleanInheritedProperties ();
5      suggestions ← $\emptyset$;
6      rules ← genRules (minSupp, minConf, T, c);
7      foreach $r \in rules$ do
           /* check Heuristic 1                                    */
8          if r.condition $\in$ schema $\wedge$
9          r.consequence $\notin$ inheritedSchema then
10             suggestions.add (r.consequence);
11     foreach $s \in suggestions$ do
12         subclasses = c.getSubclassesWith (s);
           /* check Heuristic 2                                    */
13         if |subclasses| = 1 then
14             suggestions.remove (s);
           /* check Heuristic 3                                    */
15         if |subclasses| > 1 then
16             foreach subclass $\in$ subclasses do
17                 subclass.removeProperty (s);
18     c.addSuggestions (suggestions);
```

classes in the given ontological hierarchy in a breadth-first manner, beginning with the leaves and moving up towards the root. In Lines 3 and 4 the annotated schema and inherited schema are retrieved. While the schema contains only properties that have been defined specifically for class *c*, the inherited schema also contains properties inherited from superclasses of *c*. In both cases we have excluded removal candidates, because we want to consider rules that involve properties that we define as appropriate for class *c*. Afterwards for each *c* in *classes*, rule mining is executed on all triples in *triples* belonging to instances of type *c* according to Configuration 1, based on the given minimum support and confidence thresholds in Line 6.

For each rule discovered in the process, Heuristic 1 is checked in Line 9. If it holds, the rule's consequence is added to the set of inclusion *suggestions*. As we apply the same minimum support here as in the property removal suggestion step, no property is proposed for the same class for which it has been marked for removal earlier. Note that *schema* and *inheritedSchema* contain only those properties that have not been marked when identifying property removal suggestions, i.e., every property that has been marked for removal in the previous step may appear as an inclusion candidate for some of the subclasses of c.

Having scanned *rules* for appropriate *suggestions*, the next step is to test the current class c and *suggestions* for the Heuristics 2 and 3. Thus, for each suggestion s it is checked whether there are *subclasses* of the current class c that include s in their schema or inclusion suggestions list. According to Heuristics 2 and 3, the algorithm either removes s from all definitions in subclasses (either specified or proposed) of c or from the current *suggestions* of c.

Finally, the remaining *suggestions* are added to the current class c. These inclusion suggestions can be used to extend the original class definitions.

4.3.4 Repairing Property Ranges

As noted in Section 4.2.2, the same methodology to identify misusage patterns for class properties can also be applied considering the ranges of properties. Technically, Algorithm 2 can directly be applied to identifying missing or inappropriate range definitions of properties by applying the following adaptations:

First, the notion of schema does not apply anymore. For each class instead of the schema of a class one has to consider all the properties that have a specific class in their range definition. For example, the class `:RecordLabel` is defined as the range of the properties `:recordLabel` and `:distributingLabel`. Accordingly, the inherited schema would correspond to the set of properties that have a superclass of the current class c in their range. In case of `:RecordLabel` this set would apply to properties that have `:Organisation` in their range, such as `:employer` or `:affiliation`.

Next, as we consider the range definition of a property, Configuration 1 must be replaced by Configuration 6, which mines predicates in the context of objects. Furthermore, Heuristic 1 can be omitted, because the notion of schema is not applicable here: In case of property domains, our approach proposes only to add a class to the domain of a property if the property correlates with the schema elements of that class. In case of property ranges however, there is no corresponding semantics.

4.4 Case Studies

We performed two case studies to evaluate our approach: the first case study, the analysis of the the Billion Triple Challenge 2011 dataset, shows the relevance of our approach. Considering the BTC dataset as a snapshot of the Web of Data, our analysis denotes that misusage of ontology definitions is a common and widespread problem. We present some of the results from this dataset in the following. However, as we could only speculate which ontology extensions correspond to which instance triple information in the BTC dataset, we cannot measure the quality of our approach. Therefore, we also evaluated our approach on the DBpedia dataset, for which we know the complete underlying ontology and can thus better discuss our results.

4.4.1 Billion Triple Challenge 2011 Dataset

The RDF data crawled for the Billion Triple Challenge comprises a heterogeneous snapshot of the Web of Data. There are numerous instances where existing ontological concepts are extended with new properties ad-hoc or where predicates are used without proper specification. For example, we discovered around 150 properties whose domain is `foaf:Person`, whereas the original FOAF specification declares only 16 properties for this class. Overall, we discovered 213,382 distinct classes referenced by 441,461,669 individual instances (which can be of more than one type) in the BTC 2011 corpus. Of these instances, `foaf:Person` is the most common type, accounting for 362,590,928 typed entities. To present suggestions based on a statistically relevant amount of data, we focused on the top 15 types, representing in total 405,649,267 (or around 92%) of all instances.

For a first usage analysis, we extracted instances of several common types and matched their properties with the original specification. Table 4.2 lists examples of overspecification. For example, only one instance of `foaf:Person` has a value for `foaf:plan` in the entire BTC 2011 dataset, albeit its specification in the original FOAF ontology. Some of the specified properties of `foaf:Agent` and `mo:MusicArtist` are never used at all (e.g., `foaf:tipjar` and `mo:activity_start`, respectively). Obviously, these properties might as well be removed from the class definition.

On the other hand, Tab. 4.3 lists several predicates that are commonly used for a class, but are not specified as properties in the ontology for various reasons. For example, the predicate `foaf:image` is not defined as a property for `foaf:Person` whereas `foaf:img` is (`foaf:image` might oftentimes be confused with the class `foaf:Image`).

Next, we mined association rules on instances of the common types to detect patterns that support the categorization of the class definition and data mismatch.

Table 4.2: Overspecified properties in the BTC 2011 dataset

Property	Class	Support
`foaf:yahooChatID`	`foaf:Agent`	0.000%
`foaf:tipjar`	`foaf:Agent`	0.000%
`foaf:geekcode`	`foaf:Person`	<0.001%
`foaf:plan`	`foaf:Person`	<0.001%
`mo:activity_end`	`mo:MusicArtist`	0.000%
`mo:activity_start`	`mo:MusicArtist`	0.000%

Table 4.3: Underspecified properties in the BTC 2011 dataset

Suggested Property	Class	Support
`foaf:member_name`	`foaf:Person`	5.263%
`foaf:tagLine`	`foaf:Person`	5.257%
`foaf:image`	`foaf:Person`	4.974%
`sioc:account_of`	`foaf:OnlineAccount`	82.062%
`sioc:follows`	`mo:OnlineAccount`	50.519%
`ov:sortLabel`	`mo:MusicArtist`	34.803%

Table 4.4 shows some interesting rules with confidence $\geq$ 90% among predicates that occur for instances of the types `foaf:Person` and `mo:MusicArtist`. For `mo:MusicArtist`, properties of the original ontology (e.g., `mo:remixed`) are also used frequently in combination with properties from other namespaces (e.g., `bio:event`). Hence, the latter could be added to the definition of these classes.

Table 4.5 illustrates some negative rules with $c \geq 90\%$ that we extracted from the same data. The first rule for `mo:MusicArtist` shows an obvious hint for establishing two disjoint classes for instrumental artists and vocalists. Both predicates belong to the same ontology source and do not have contradicting meanings, but in the BTC data the set of vocalists and the set of instrumentalists are nearly disjunct. The negative association rule between `foaf:homepage`, `mo:myspace`, and `foaf:page` might imply that different data publishers use different properties for describing the same resource (i.e., an artist's web page). Looking at the rule examples for `foaf:Person`, one can recognize that most negative associations are between synonyms or similar resources such as `foaf:name` and `foaf:nick`, `foaf:homepage` and `foaf:weblog`, or `foaf:image` and `foaf:img` (as pointed out earlier, `foaf:image` is misused as a predicate very often). We discovered many negative association rules between predicates from different namespaces (e.g., 6979 rules that hold 0.2% minimum support and 80% minimum confidence among facts from `mo:MusicArtist`) , although the associated instances are of the same type.

Table 4.4: Association rule examples with $c \geq 90\%$ and $s \geq 0.2\%$

foaf:Person		
foaf:img	→	foaf:nick
foaf:gender	→	foaf:weblog
foaf:mbox_sha1sum	→	foaf:homepage
foaf:weblog	→	foaf:member_name
mo:MusicArtist		
mo:image	→	foaf:depiction
mo:myspace	→	mo:musicbrainz
dc:description	→	foaf:homepage
mo:remixed	→	bio:event
mo:myspace	→	mo:wikipedia
(foaf:name, foaf:page, mo:member_of)	→	mo:musicbrainz

Table 4.5: Negative rule examples with $c \geq 90\%$ and $s \geq 0.2\%$

foaf:Person		
foaf:weblog	→	¬ foaf:homepage
foaf:image	→	¬ foaf:img
foaf:name	→	¬(foaf:nick, foaf:gender)
mo:MusicArtist		
mb:isInstrumentalArtistOf	→	¬ mb:isVocalistOf
foaf:page	→	¬ (foaf:homepage, mo:myspace)

As mentioned earlier, we consider the crawl of the Billion Triple Challenge a unique snapshot of the Web of Data. Therefore, in contrast to individual data sources, the rules outlined above give some indication about a broad misconception of certain ontological structures. Instead of single data publishers misusing defined vocabularies, this might hint at a required design overhaul of these vocabularies. For more results of our experiments, please visit `http://www.hpi.uni-potsdam.de/naumann/projects/btc/btc2011.html`.

4.4.2 DBpedia Dataset

To further assess the quality of our ontology re-engineering proposals, we also applied our approach to the DBpedia dataset and corresponding ontology, and evaluated the resulting suggestions. In contrast to the previously discussed FOAF and Music ontologies, the DBpedia ontology has been created posterior to the data it corresponds to. The ontology is manually generated using Wikipedia infobox templates [Bizer et al., 2009b], and evolves over time as the infobox templates are changed[1]. Our algorithm generates useful property suggestions even if applied to the hand-curated DBpedia ontology.

We performed our evaluation on the DBpedia 3.6 and DBpedia 3.7 datasets along with the respective DBpedia ontology versions. In Table 4.6, we list the total amounts of (unique) triples and subjects in the individual datasets as well as the total amounts of unique classes and properties in the ontologies. We applied our algorithm to identify improvement suggestions for property domains as well as property ranges.

Table 4.6: Number of distinct values occurring in DBpedia 3.6 and 3.7 data

DBpedia	#Triples	#Subjects	#Classes	#Properties
3.6	17,518,364	1,638,746	272	1,335
3.7	13,794,426	1,827,474	319	1,643

Class Definition Suggestions

By applying Algorithm 2 to improve class definitions or rather property domain definitions, 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 4.7 shows sample results of overspecification in DBpedia 3.7. Some of the removal suggestions can be moved to a more suitable subclass, Tab. 4.8 presents such an alternative allocation of the bottom five properties in Tab. 4.7. Clearly, the proposed assignment is more appropriate than the actual specification, as the suggestion support is orders of magnitude higher than the original support. While for some properties the suggestion support still seems low (e.g., `:countySeat`), instances of these classes still include more than 90% of the occurrences of the properties among the originally assigned parent class.

For underspecification, we applied Algorithm 2 (*minSupp*: 1%, *minConf*: 70%) to DBpedia 3.6 and DBpedia 3.7. A higher *minSupp* threshold results in more removal suggestions and less inclusion suggestions. A higher *minConf* threshold reduces only the number of inclusion suggestions. We evaluated all suggested properties for the classes of the ontology manually by letting two computer scientist labelers label whether their assignment to a specific class was:

- *Useful*: Properties were appropriately assigned to a class.
- *Not useful*: Predicates were inappropriately assigned to a class.

[1]See `http://wiki.dbpedia.org/Changelog` for a complete changelog

Table 4.7: Overspecified properties for DBpedia 3.7

Property	Class	Support
`:scottishName`	`:Settlement`	0.000%
`:distanceToEdinburgh`	`:Settlement`	0.021%
`:waistSize`	`:Person`	0.013%
`:philosophicalSchool`	`:Person`	0.202%
`:countySeat`	`:PopulatedPlace`	0.831%
`:anthem`	`:PopulatedPlace`	0.147%
`:depth`	`:Place`	0.723%
`:numberOfGraduateStudents`	`:EducationalInstitution`	0.300%

Table 4.8: Pushed properties for DBpedia 3.7

Suggested Property	Class	Support
`:philosophicalSchool`	`:Philosopher`	76.225%
`:countySeat`	`:AdministrativeRegion`	9.470%
`:anthem`	`:Country`	18.730%
`:depth`	`:Lake`	33.698%
`:numberOfGraduateStudents`	`:College`	93.590%

- *Undecided*: Labelers could not determine the appropriateness.

Overall, the majority of the suggestions have been labeled as useful, including the pushed properties mentioned in Tab. 4.8. Some of our proposed properties for the DBpedia 3.6 dataset have indeed been included in the DBpedia 3.7 ontology, such as `:numberOfEpisodes` and `:numberOfSeasons` for the class `:TelevisionShow`, thus validating our results.

Table 4.9: 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

Table 4.9 illustrates the amount and quality of class property suggestions for DBpedia 3.6 and 3.7. Suggestions marked as undecided are those for which the labelers 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). Table 4.10 lists the origins of the proposed properties by indicating for how many of them…

- …*no domain* was specified,
- …the original domain was a sub-/superclass and the property has been *pushed* up/down to a more appropriate super-/subclass,
- …a completely *different* domain was assigned, or
- …a *synonymous* property was specified, but used only infrequently or never at all (e.g., for `:Person`, `:Person/weight` is specified, `:weight` is suggested).

Table 4.10: Origins of suggested underspecified properties in DBpedia 3.6 and 3.7

DBpedia	Total	No Domain	Pushed Up/Down	Different Domain	Synonymous
3.6	283	206	17	12	48
3.7	317	225	18	17	57

For DBpedia 3.6 we discovered that of the 283 suggested properties 206 had no specified domain, 17 were pushed up or down from a sub- or superclass, 12 had completely different domains, and 48 had synonymous specified properties. Of the 317 suggested properties in DBpedia 3.7, 225 had no domain, 18 were pushed, 17 had different domains, and 57 had synonymous specified properties. For most of our suggestions no domain was set, making it easy for those properties to be assigned to a specific class. Note that we considered only properties from the DBpedia ontology namespace, but properties from other namespaces (such as the FOAF or DBpedia property namespace) might also be valid suggestions. Overall, our evaluation shows that data-driven suggestions lead to useful results.

Property Range Suggestions

We also applied our approach in the adapted fashion described in Section 4.3.4 to identify improvement suggestions for property range definitions. It is important to note that this time we omit Heuristic 1 and therefore need only to set the support parameter, which was set to 1%.

In the first phase of our approach, we identified 267 cases for both DBpedia versions 3.6 and 3.7, where the class specified as the range of a property was not significantly represented by instances of the data. In fact 117 property ranges from DBpedia 3.6 and 104 property ranges from DBpedia 3.7 had not even one member of the specified class in their range. The rest were replaced with new range suggestions. Table 4.11 illustrates examples of properties and classes that have been identified as inappropriate for their range. For the first five properties where at least some fraction of the class instances occurred as range values of the specific property, our approach proposed new classes as illustrated in Table 4.12. Comparing the results in Table 4.12 with Table 4.8 we can see that the new proposals for range definitions lead only to marginal support improvement. This shows that domain definitions of properties are usually more coherent than range definitions.

Table 4.11: Overspecified properties for DBpedia 3.7

Property	Class	Support
`:canonizedBy`	`:Person`	0.003%
`:governor`	`:Person`	0.001%
`:highestPlace`	`:PopulatedPlace`	0.0001%
`:heir`	`:Person`	0.01%
`:discoverer`	`:Person`	0.004%
`:architecturalBureau`	`:Company`	0.0%
`:polishFilmAward`	`:Award`	0.0%
`:grammyAward`	`:Award`	0.0%

Again we classified all inclusion suggestions based on the classification we presented previously. Table 4.13 illustrates our results. In general, we had many more suggestions for property ranges than for domains (See Table 4.9). The reason is threefold. First, range definitions are more often missing than domain definitions. Second, range definitions usually contain more classes than domain definitions, which again shows that domain definitions are more coherent than range definitions. Finally, many range definitions are originally set as too general, having not enough support in the instances. In such cases it happens that several small

Table 4.12: Pushed properties for DBpedia 3.7

Property	Suggested Class	Support
`:canonizedBy`	`:Cleric`	59.8%
`:heir`	`:Monarch`	1.2%
`:governor`	`:Politician`	5.1%
`:highestPlace`	`:Mountain`	42.4%
`:discoverer`	`:Scientist`	26.5%

subclasses are suggested to serve as the range of the same property. For example the property `:monarch`, which originally had the range `:Person`, is now suggested to have the classes `:Monarch`, `:BritishRoyalty`, and `:ChristianBishop` in its range definition.

Table 4.13 illustrates the total numbers for range suggestions on DBpedia 3.6 and 3.7. Compared to the results in Table 4.9, the numbers show that many of the inclusion suggestions for range definitions are not useful. 44% of the usage patterns of a property in DBpedia 3.6 and xx% in DBpedia 3.7 are not intuitively reasonable.

Table 4.13: Range suggestions for DBpedia 3.6 and 3.7

DBpedia	Total	Useful	Not Useful	Undecided
3.6	794	437 (55%)	268	89
3.7	953	551 (58%)	295	106

One reason for the different scale of quality results with regard to property domains and ranges could be that users intuitively use reasonable instances with regard to the domain of properties this is not true for their range. Given a subject, a very often user chooses the reasonable predicates to describe the subject, while when completing a statement the user has very different interpretations of a property's actual meaning. Another reason might be that specific properties that are needed by the user, such as `decoratedFor`, are missing. For example, the properties `:award`, `:grammyAward`, `:olivierAward`, which originally have the class `:Award` as the specified range, do not occur with a specific award, such as `Grammy Award`, in the dataset but instead with the notable works that lead to that award. In DBpedia 3.7, we encounter 396 statements where the property `:award` specifies the awarded work of an artist. Given those results, knowledge base curators have to identify the trade-off between adapting the ontology to the data and changing the dataset to match the ontology.

The runtime of our algorithm is in the order of a few hours on a MacBook pro with a 2.66 GHz Intel Core 2 Duo processor and 4 GB DDR3 RAM and a remote database (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.

4.5 Summary

We identified and described two misconceptions of the vocabulary definition regarding its application to real-world data: over- and underspecification. To cope with these challenges, we presented an automated approach that facilitates ontology re-engineering by suggesting the removal or incorporation of properties for given ontology classes.

Based on the mining configuration methodology, we applied association rule mining and frequency analysis to evaluate the usage of an ontology in real-world RDF data and to suggest possible modifications to the ontology's definition. Our approach can easily be adapted to identify misconceptions in property ranges, too. An ontology engineer can use these suggestions to revise the specification, thereby improving the intuitiveness of the ontology and aiding its propagation.

We evaluated our approach on both the heterogeneous BTC 2011 dataset and the different ontologies used therein as well as the more homogeneous DBpedia dataset along with the DBpedia ontology. As illustrated in Section 4.4, there are significant mismatches between the intended use of certain well-known ontology specifications and how they are employed. These might be caused by illegitimate use of the vocabulary by data publishers, ontology evolution, or general design flaws in the vocabulary, amongst other things. By examining these differences, we were able to identify different causes:

1. Illegitimate use of the vocabulary by data publishers (e.g., because of a lack of knowledge of specification details), especially with regard to the property ranges.

2. The ontology has evolved over time and needs to be revised as class definitions are no longer suited for current real-world data.

3. General design flaws in the vocabulary, including term ambiguity or confusing schema term definitions.

One of the most common phenomenons that is caused by ontology misusage are synonymously used predicates. Being not aware of the existence of such properties prevents the effective usage of a knowledge base in order to find relevant information. In the next chapter, we present a rule-based approach to automatically identify pairs of synonymously used predicates.

Chapter 5

Synonym Analysis for Predicate Expansion

In the previous chapter, we showed that existing knowledge bases suffer from inconsistency, which is partly caused by the fact that data publishers use custom properties as predicates in their statements instead of properties that were specified in the corresponding ontology. Evaluations on the DBpedia data set showed that some of the mismatches occurred, because predicates synonymous to a property defined by the ontology were deliberately used, e.g., city or location instead of locationCity. Of course two synonymous predicates may have been defined on purpose for two disjoint semantical roles, yet as they have been used in substitution of each other, the data consumer has to cope 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 5.1. 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. To support the usage of LOD, it is useful to identify the occurrence of synonymously used predicates to immediately expand a query at hand or at least to notify a user about the existence of alternative property candidates.

In the field 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 as well as ontology and

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

Table 5.1: 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.

schema matching across multiple ontologies have already been extensive subject of research [Volz et al., 2009; Suchanek et al., 2011; Li et al., 2009; Udrea et al., 2007]. However, the discovery of synonymously used predicates within a single knowledge base has not received attention yet.

The discovery of synonymous predicates also support data creation: A user might be able to avoid creating duplicate triples. However, avoiding the creation of semantically equivalent, hence redundant, triples in the data set might be much more cumbersome. Knowledge of synonymously used predicates in a dataset can be applied to notify an editing user when she creates a new redundant triple while a semantically equivalent triple is already available.

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. The traditional synonym discovery in unstructured data, such as web documents, needs to apply natural language processing rules.

Last but not least, for many data sources meta-data is only poorly provided. Identifying synonymously used predicates can support the evaluation and the 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 by different data publishers. So a data-driven approach is necessary to dissolve the existing synonym dependencies.

Applying our system of mining configurations, we present an approach, introduced in [Abedjan and Naumann, 2013b], for discovering predicate pairs that substitute each other in the data and are good candidates for query expansions. To differentiate our contributions from previous work, we first discuss existing approaches on synonym discovery, schema matching, and query expansion with special focus on open RDF data. In Section 5.2, we formulate our definition of synonymously used predicates in RDF data. Then we describe our mining-configuration-based approach in Section 5.3 and evaluate our approach and strategies in Section 5.4.

5.1 Related Work

The identification of synonymously used predicates in RDF knowledge bases is related to several fields. We first outline state-of-the-art on data-driven approaches for discovering synonyms. Then we discuss existing approaches for the purpose of query expansion to position our contribution for this use case and finally we differentiate our work from the classical field of schema and ontology matching.

5.1.1 Synonym Discovery

Most of the 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 [Baroni and Bisi, 2004; Wei et al., 2009; Turney, 2001] or paraphrases [Grigonytė et al., 2010]. The intuition behind these approaches is that synonymous words significantly co-occur in the considered text units [Harris, 1954]. So, they calculate the ratio of real co-occurrence of two terms and the independent occurrence of each term. It is notable that for these approaches there are already known candidate pairs, which only have to be validated. In our scenario this assumption does not hold, because we also have to discover and retrieve the candidate pairs.

While Turney et al. concentrate on globally valid synonyms [Turney, 2001], Baroni et al. concentrate on synonyms in a subdomain of the English language (nautical terminology) [Baroni and Bisi, 2004]. Both approaches compute the probability of two terms being synonyms based on the mutual information of both terms that is estimated by the number of search results of the AltaVista search engine. Wei et al. address context sensitive synonym discovery by looking at co-clicked query results [Wei et al., 2009]. 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 web tables [Cafarella et al., 2008]. The authors introduce the metric *Syn*, which can be used in order to identify synonyms among table attributes. However, their approach has an important restriction: they assume a context attribute that has to co-occur with synonym pair candidates and ensures that only attributes of a certain domain context are considered as synonym candidates. Furthermore, they ignore instance-based techniques as they process only extracted table schemata. In Section 5.3.1, we discuss the *Syn* function in more detail, as we adapt it to our scenario.

5.1.2 Query Expansion

Research on query expansion includes stemming techniques, relevance feedback, and other dictionary-based approaches [Baeza-Yates and Ribeiro-Neto, 1999]. On their technical level, the approaches do not apply to our SPARQL scenario as we do not retrieve documents, but structured entities. Most notable work in the area of SPARQL relaxation is a query expansion approach based on language models [Elbassuoni et al., 2011, 2012]. Our approach is based on association rules and it benefits from a more simplistic model so that we were able to process large datasets, such as DBpedia, in minutes.

There has also been research on relaxing queries on structured data. Zhou et al. present an approach to extend database queries with malleable schemas [Zhou et al., 2007]. For that purpose they identify overlapping properties, such as *name* and *firstname*, by applying duplicate-driven schema-matching [Bilke and Naumann, 2005]. The intuition of their approach is very similar to ours, however we do not rely on duplicates in the data. Instead we follow a different schema matching intuition by combining instance-based and schema-based filtering to identify such properties in RDF data. We address related schema matching techniques in the following subsection. Finally, Koudas et al. present an approach to relax numerical selection and join attributes by extending the selection ranges [Koudas et al., 2006]. Our approach is orthogonal to this line of research as we want to extend queries results by adding new ranges from other similar/synonymous attributes.

5.1.3 Schema Matching

We already identified that some query relaxation techniques, such as [Zhou et al., 2007], that deal with structured data are based on schema matching techniques. Before identifying technical overlaps of our solution with state-of-the-art schema matching approaches, it is important to note the difference between schema matching and synonym discovery. 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 [Rahm and Bernstein, 2001], we would classify our approach as a mixture of an instance-based and a schema level matching algorithms.

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 [Doan et al., 2001; Gottlob and Senellart, 2010; Naumann et al., 2002; Bilke and Naumann, 2005]. Choosing features to match values of an attribute is cumbersome and domain dependent. Furthermore, 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. As experiments show, the mining-based approach is much more efficient.

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 [Li and Clifton, 2000]. Our approach focuses on mining-based features on graph level in order to discover synonymously used predicates.

5.2 Synonymously used Predicates

In this work, we explicitly talk about *synonymously used* predicates instead of *synonym* predicates and define them as follows:

Definition 4 *Two predicates p_1 and p_2 are synonymously used, $p_1 =_{SU} p_2$, when they are interchangeably used in a given dataset to denote equivalent information.*

Accordingly, for each predicate $p \in P$ (P is the set of all predicates that occur in the dataset) we denote the set of synonymously used predicates as

$$SU(p) = \{q \in P | q =_{SU} p\}.$$

While two terms are deemed to be synonyms, when they have the same meaning, Definition 4 is more relaxed: We not only avoid linguistic discussions about synonyms, but we also identify predicate pairs that substitute each other in an RDF dataset without being actual synonyms. 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`. Another example refers to pieces of work, such as songs that are linked to the songwriter. We encounter examples that are linked via the predicate `author` as well as `lyrics` to the songwriter of the song. Although `author` and `lyrics` are not synonyms in a linguistical sense, both serve the same purpose, i.e., are synonymously used.

To identify that two predicates p_1 and p_2 are being synonymously used we adhere to the following two intuitions:

1. The predicates occur interchangeably, i.e., a given entity usually does not include both properties in its schema.

2. The predicates have an overlapping range of object values.

If a user is interested in all movies with the movie star *Al Pacino*, he wants to retrieve the set M of all movies where *Al Pacino* acted in, no matter which predicate denotes that information. That means $M = \{s|(s, p, \texttt{Al Pacino}) \in KB \wedge p =_{SU} \texttt{starring}\}$. The first intuition directly applies to the problem that query results may be incomplete without considering synonymously used predicates. The interchangeable occurrence of p and `starring` contributes to the number of elements in M. If properties co-occur for entities, they either are not synonymously used or they are redundant synonyms and do not contribute to the completeness of our query result M. Our second intuition ensures that we identify similar properties.

In the next section, we describe how we identify all predicates pairs $p_1 =_{SU} p_2$ in a knowledge base by applying our mining-based methodology.

5.3 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 pairs $p_1, p_2 \in P$ with $p_1 =_{SU} \; p_2$ the expansion candidates of a given predicate $p \in P$ can easily be retrieved by selecting all synonymously used predicates $SU(p) = \{p' \in P|p =_{SU} \; p'\}$.

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. 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 can be derived from existing schema-level and instance-based schema matching techniques.

5.3.1 Schema analysis

Configuration 1 enables us to do frequency analysis and rule discovery per entity. In Chapters 3 and 4, we already showed that positive rules between predicates can be used for auto-completion and re-validating existing ontologies. To discover synonymously used predicates, we follow 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. In order to compute negative correlations for a set of candidate pairs, we extended our FP-Growth [Han et al., 2000] implementation to deliberately retrieve itemset frequencies even if they are not frequent. 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 scoring functions we tried at schema level.

Confidence aggregation

The confidence *conf* of the rule $p_1 \rightarrow \neg p_2$ describes the probability c% of predicate p_2 to not occur for the same entity where p_1 occurs. We refer to these rules as negative rules as introduced in Section 2.1. If p_2 was a rare predicate that, however, occurs always with p_1, $\mathit{conf}(p_1 \rightarrow \neg p_2)$ might be considerably high however $\mathit{conf}(p_2 \rightarrow \neg p_1)$ 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 dataset. For example, if two predicates exclusively occur only once in a set of ten transaction any confidence aggregation on negative rules will result in the highest score 100% and the same would apply if both predicates exclusively occured five times in the same set. Clearly, the significance of the exclusivity for the case where both predicates occur more often but exclusively is much higher. To capture the significance of such exclusivity, one can directly apply the correlation coefficient. We apply the formula given in [Antonie and Zaïane, 2004], which measures the linear relationship between the two predicates:

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

where N denotes the total number of baskets in the mining configuration, which, for Configuration 1, is equivalent to the total number of entities $|S|$ in the dataset. For ranking purposes, want to have positive scores on negative correlations. Therefore, we introduce the reversed correlation coefficient (RCC), by reversing the sign of *cCoeff*(X, Y):

$$RCC(X, Y) = -cCoeff(X, Y)$$

***Syn*-function**

Cafarella et al. introduce the *syn*-function for synonym discovery across different webtables [Cafarella et al., 2008]: Their assumptions are also that synonyms never occur together and have the same odds in occurring with other attributes. In case of LOD 'never' is a too strong assumption for open RDF data. That is why, we relaxed the constraint by filtering candidate pairs that are not negatively related based on the *RCC* score. Furthermore, the *Syn* score

$$Syn(X, Y) = \frac{p(X)p(Y)}{\epsilon + \sum_{z \in P} (p(z|X, C) - p(z|Y, C))^2}$$

is based on a set of context attributes C. Unfortunately the authors do not mention how to choose this context attribute, if domain knowledge is not given. Nevertheless, the intuition behind their function can also be applied to our scenario for synonymously used predicates. 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 additionaly considering the range of predicates.

5.3.2 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 p_1 and p_2, one would look at the ratio of overlaps depending on the total number of values of such a predicate:

$$\frac{{}^{p_1}O \cap {}^{p_2}O}{min({}^{p_1}O, {}^{p_2}O)}$$

We apply a more efficient range-content filtering approach (RCF) based on mining configurations (see Chapter 2).

Configuration 6 constitutes a mining scenario where each transaction P^o is defined by a distinct object value o. 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. 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 loss of quality. Similar to the schema analysis strategy, also the range-content filtering approach 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 combining both approaches, this candidate pair can be pruned looking at their exclusion rate per entity during schema analysis.

5.3.3 Range Structure Filtering

In some scenarios, value range content filtering might not be the most appropriate technique as it requires two synonymously used 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 at general string or token characteristics, such as string length and character distributions [Doan et al., 2001; Naumann et al., 2002]. As the goal of this work 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. Thus 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 ignored the general types, which are also the most frequent ones, and 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. Experiments showed that weighted Jaccard similarity is 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. Literals and objects without a type assignment, e.g., dates and other numerical values, have been handled as unknown types, whereas no two unknown types are equal. Considering unknown type as equal types will result into many false positives as the cardinality of the unknown type component in the type vector will become too large, while it covers many completely unrelated objects.

5.3.4 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 the 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.

The mentioned order of operations has two advantages: as retrieving negative correlations and type vectors is time-consuming, it is reasonable to perform both on reduced candidate sets instead of using them on the complete dataset to retrieve the 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. It is also important to note that type range filtering can only be applied to datasets for which the type information is available. In our experiments, the type filtering approach could only be applied to the DBpedia datasets, and for that dataset it actually did not contribute to the precision of the expansion candidate discovery on top of range-content filtering.

5.4 Evaluation

We evaluated our approach on discovering synonymously used predicates with regard to the precision and recall of generated expansion candidates on multiple datasets. Table 5.2 shows the data sets with the corresponding numbers of distinct triples, subjects, predicates, and objects. Because DBpedia contains data from different domains, we again performed additional 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 the overall precision results on multiple data sets. Last, we illustrate the efficiency gain of our mining configuration based overlap discovery method towards the standard value-overlap approach.

Table 5.2: 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

5.4.1 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 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, just according to $=_{SU}$. 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 Figure 5.1. For this purpose we sorted all pairs twice. For the range-content filtering curve, we sorted the results by their support and for the type filtering curve we sorted with regard to the weighted Jaccard similarity. Considering the data point at 100% recall, both approaches perform better than a random approach, which results in 0.8% precision. However, the precision of the range-content filtering method is on all recall levels better than the precision achieved with range-type filtering. In fact, performing range type filtering on top of the range-content filtering method did not lead to better filtering, which makes the range-type filtering approach obsolete.

We now examine the results by using the schema analysis component on the retrieved candidates. For that purpose, we chose the two different support thresholds 0.1% and 0.01% for the content filtering part, which resulted in precision/recall scores of 52%/28% and 22%/98% respectively as illustrated in Figure 5.1. Figures 5.2a and 5.2b illustrate the ranking improvement of the algorithm using the schema scores. At the support threshold of 0.01% the range content filtering

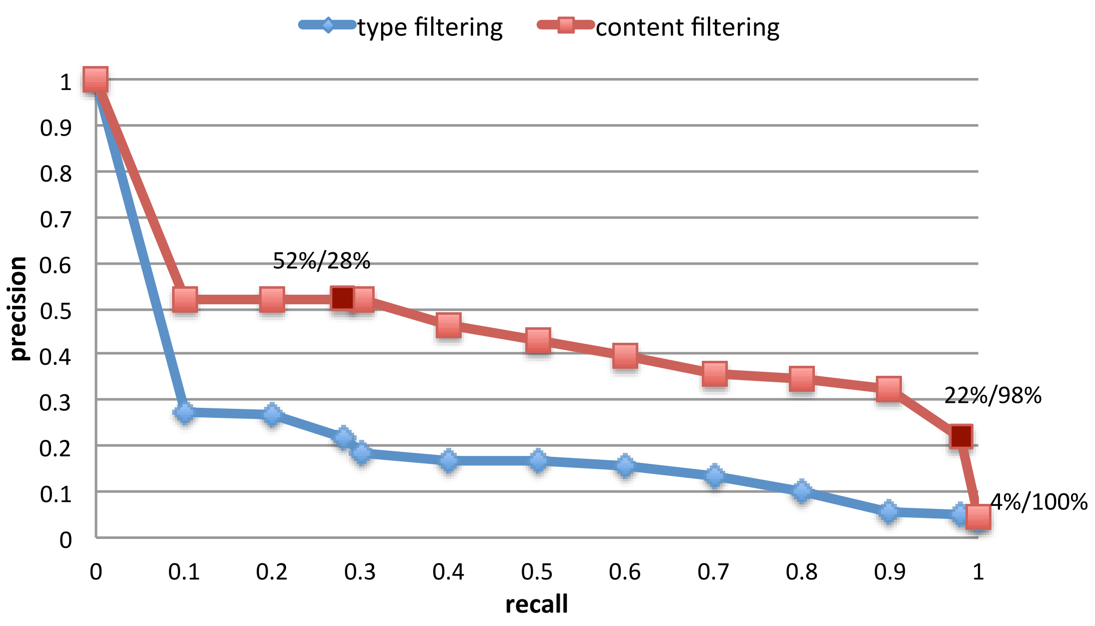
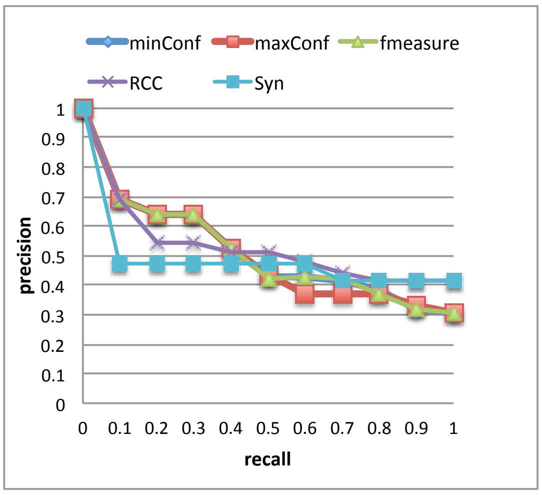


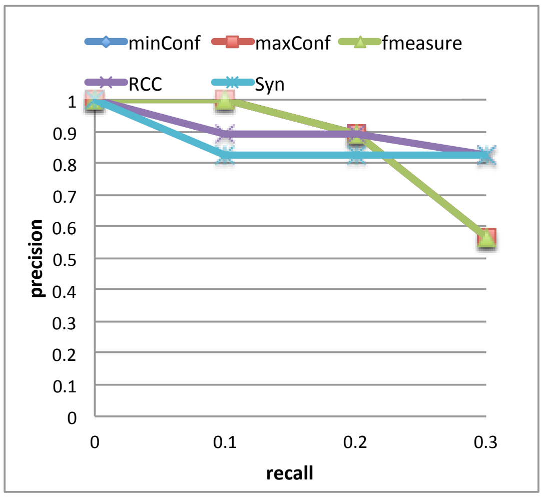
Figure 5.1: Precision recall curve for the filtering methods

achieved 22% precision and 98% recall. Figure 5.2a shows that all schema scores result in better precision than range-content filtering on this support level. In particular, on lower recall levels the precision is significantly higher than the results in Figure 5.1. 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 and are not being considered. The results in Figure 5.2b also show a clear precision enhancement through schema analysis methods. However, the only difference to the previous set of experiments is that at the highest possible recall level of 28%, only the RCC score leads to better results. Summing up both experimental settings, it can be observed that at high recall levels the RCC score performs better than all other scoring functions. The *Syn* function performs worse than all approaches on all recall levels. In general, one could argue that applying any of the negative correlation scores leads to better results than sole range-content filtering.

Table 5.3 denotes the top-10 matched predicate pairs on the DBpedia Work dataset for the different schama analysis scores. Note the top-10 results from minimum confidence and the aggregated F-Measure confidence are exactly the same. In particular, the first eight results are exactly the same for all confidence aggregations because the items never co-occurred. That means minimum confidence and maximum confidence of the negative rules were both 1.0. Considering the top 10 results for the *Syn* function, one can clearly identify that some of the matched pairs, such as `director` and `author`, are harder to justify although it is



(a) 0.01% support



(b) 0.1% support

Figure 5.2: Precision recall curve of schema scores on the Work dataset

not very difficult to create a link between those two.

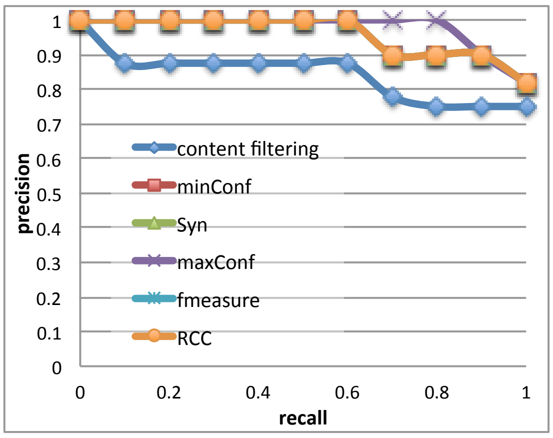


Figure 5.3: Precision recall on Magnatune with 0.0% minimum support

We performed similar experiments on the Magnatune dataset. Regarding the results in Figure 5.3, we can observe very high precision values even with range content filtering. However, even at a support threshold of 0.0%, the all schema-based scoring functions 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.

Table 5.3: Top 10 results from the DBpedia 3.7 Work dataset ranked by different schema analysis scores

Rank	Max confidence	Min/F-Measure confidence
1	`artist starring`	`artist starring`
2	`writer author`	`writer author`
3	`starring guest`	`starring guest`
4	`artist musicComposer`	`artist musicComposer`
5	`writer creator`	`writer creator`
6	`recordDate completionDate`	`recordDate completionDate`
7	`writer executiveProducer`	`writer executiveProducer`
8	`musicComposer composer`	`musicComposer composer`
9	`musicComposer composer`	`releaseDate publicationDate`
10	`releaseDate latestReleaseDate`	`producer author`

Rank	RCC	Syn
1	`artist starring`	`artist writer`
2	`artist writer`	`artist starring`
3	`artist musicComposer`	`artist musicComposer`
4	`producer author`	`producer author`
5	`writer author`	`producer creator`
6	`director author`	`producer composer`
7	`starring author`	`director author`
8	`artist composer`	`starring author`
9	`writer creator`	`artist composer`
10	`releaseDate publicationDate`	`writer author`

5.4.2 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, here we use the absolute value 0.0 as the threshold.

Comparing both Tables 5.4 and 5.5, one can see the precision improvement by leveraging the support threshold for range-content filtering (RCF in the tables). Furthermore, one can observe that all schema scores behave very similarly. 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.

Table 5.4: 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 Org.	**33%**	32%	32%	31%	32%	32%	412

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 always reasonable to perform the algorithm on each domain fraction of the data set separately if this distinction is available. For instance, performing the

Table 5.5: 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

experiments on entities of the more specific type Actor, which is a subclass of Person, we achieved much better precision, e.g., range-content filtering and RCC achieved 65% and 87% respectively.

5.4.3 Efficiency Analysis

We stated that our RCF approach for discovering value overlaps using Configuration 6 (see Sec. 5.3.2) is more efficient than pairwise comparison of predicates. Table 5.6 illustrates some runtime comparisons; we aborted runs after three hours. Our mining-based range-content filtering approach is always 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 less than 8 minutes for each presented dataset at a minimum support of 0.1% for range content filtering and less than 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.

Table 5.6: 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

5.5 Summary

In this Chapter, we presented several strategies for automatically discovering synonymously used predicates that can be employed to expand or relax SPARQl queries. We showed that a combination of the Configurations 1 and 6 neatly retrieves a set of reasonable predicate pairs in an appropriate time frame.

We showed the strengths and weaknesses of the strategies on different datasets, proposing a combined algorithm based on range-content filtering and schema analysis. The evaluation showed that the combined algorithm performs well on data that contains only subjects of one domain, but produces more false positives in datasets where the subjects represent entities of many different types. If some kind of clustering or taxonomy is available, it is advisable to run the algorithm on each part of the data according to the given types or classes separately to achieve the best possible results. 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.

Chapter 6

Integrating Mining Configurations into ProLOD++

In the previous sections, we presented several applications, such as auto-completion, fact generation, ontology alignment, and synonym analysis, which are all intended to improve RDF data. In order to provide the functionalities to the Semantic Web community and to demonstrate their capabilities, we integrated adapted versions of the described algorithms into the browser-based profiling tool ProLOD++ [Abedjan et al., 2014].

Profiling tools enable data consumers and data engineers to examine new and large datasets providing appropriate interfaces and interesting meta-data [Naumann, 2013]. For profiling relational datasets, there are already many commercial tools, such as IBM's Information Analyzer, Microsoft's SQL Server Integration Services (SSIS), Informatica's Data Explorer, Data Tamer [Stonebraker et al., 2013], and some research prototypes, such as [Kandel et al., 2012]. However none of these tools were designed to profile RDF data. Open RDF data, such as LOD, has a very different nature and calls for specific profiling and mining techniques. Current tools to work on RDF data are limited to graph visualization and editing: LODlive[1] is a browser-based tool to browse and search in RDF datasets. RDF Pro[2] is a suite for visual editing of RDF data. LODStats [Auer et al., 2012] is a stream-based approach for gathering comprehensive statistics about RDF datasets. Finally, RelFinder [Heim et al., 2010] is a web-based tool to interactively discover relationships between entities on the Web of Data.

ProLOD, the predecessor of ProLOD++, was among the first online tools for profiling RDF data and was already able to generate meta-data, such as simple statistics (resource frequencies, value patterns, value distributions), positive and negative association rules in the manner of Configuration 1 (schema analysis), and

[1] `http://en.lodlive.it/`

[2] `http://www.linkeddatatools.com/rdf-pro-semantic-web`

inverse predicates [Böhm et al., 2010]. Furthermore, it provided a schema-based clustering and labeling approach to identify subgroups of entities.

By integrating the mining-based approaches described in this thesis, we significantly upgraded the older profiling tool ProLOD. With ProLOD++, we not only provide a proof-of-concept; but clearly go beyond the existing standards of available tools with regard to meta-data generation by providing sophisticated profiling analysis for RDF data. In order to build an interactive analysis tool, we had to adapt and slacken some of the approaches. The ProLOD and ProLOD++ were both implemented in collaboration with the respective co-authors of the two papers. In the following we first give an overview of ProLOD++ general architecture and then illustrate the new mining-based features.

6.1 ProLOD++ Overview

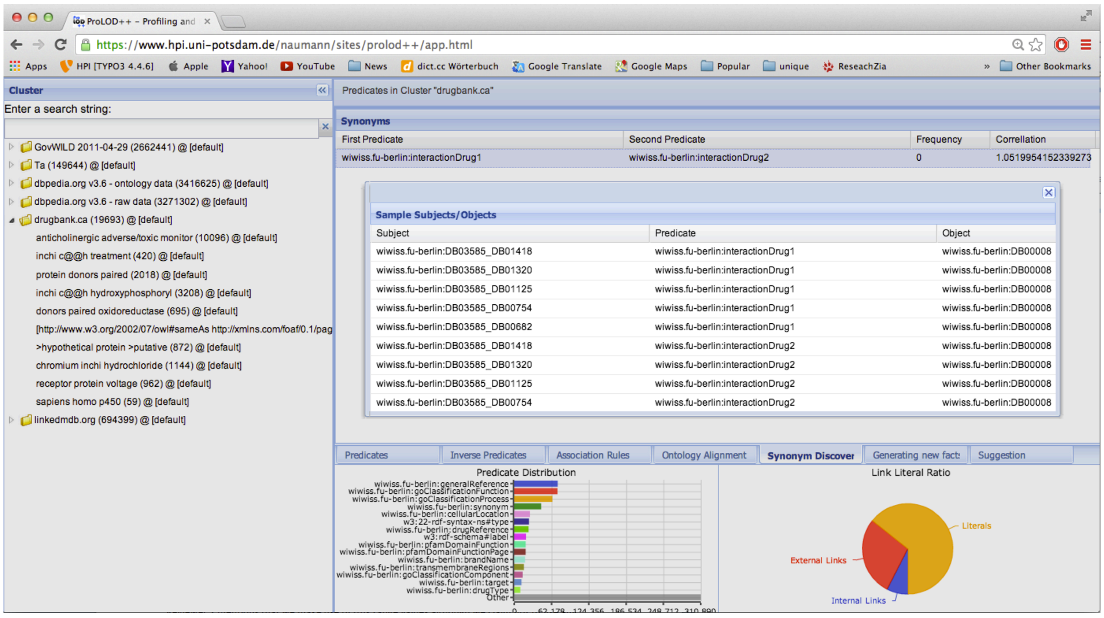
ProLOD++ is a browser-based profiling tool based on a client-server architecture. It is implemented in Java using the Google Web Toolkit (GWT)[1], which adheres to the Model View Controller pattern. Figure 6.1 illustrates the graphical view of the tool. While on the left side of the interface the data source and its subclusters or subcategories, which are organized as a tree, can be browsed, the right frame illustrates the analysis results. In Figure 6.1, we can see that the synonym discovery tab is selected.

Figure 6.2 illustrates the overall component architecture of ProLOD++. The client side of the tool is accessed via a web browser, where profiling results and data samples are visualized. Administrative tasks, such as the import/deletion of data, re-clustering and cluster re-labeling can also be initiated using the graphical user interface. Each of these tasks is then performed on the server side. Some of the tasks, such as the initial clustering, labeling, and pattern generation are performed during the import of a new dataset.

The data is stored in a commercial relational database in the form of a star schema with a maintable, which contains only IDs for each resource and many mapping tables that resolve the IDs. Each RDF triple corresponds to one row of the maintable and the resource IDs of the triple are stored within the three columns, subject, predicate, and object, respectively. Compared to complete resource URIs, the concise ID representation of the resources enables profiling tasks to work more efficient in order to provide an online system with short response time for a user. IDs are resolved as soon as a mining task is finished and the results can be sent to the client.

Once a mining-based task, such as synonym analysis or fact generation, is requested, the required data is retrieved via a data loader that connects associa-

[1] `http://www.gwtproject.org/`



- GovWILD 2011-04-29 (2662441) @ [default]
- Ta (149644) @ [default]
- dbpedia.org v3.6 - ontology data (3416625) @ [default]
- dbpedia.org v3.6 - raw data (3271302) @ [default]
- drugbank.ca (19693) @ [default]
  - anticholinergic adverse/toxic monitor (10096) @ [default]
  - inchi c@@h treatment (420) @ [default]
  - protein donors paired (2018) @ [default]
  - inchi c@@h hydroxyphosphoryl (3208) @ [default]
  - donors paired oxidoreductase (695) @ [default]
  - [http://www.w3.org/2002/07/owl#sameAs http://xmlns.com/foaf/0.1/pag
  - >hypothetical protein >putative (872) @ [default]
  - chromium inchi hydrochloride (1144) @ [default]
  - receptor protein voltage (962) @ [default]
  - sapiens homo p450 (59) @ [default]
- linkedmdb.org (694399) @ [default]

Synonyms

First Predicate	Second Predicate	Frequency	Correlation
wiwiss.fu-berlin:interactionDrug1	wiwiss.fu-berlin:interactionDrug2	0	1.0519954152339273

Sample Subjects/Objects

Subject	Predicate	Object
wiwiss.fu-berlin:DB03585_DB01418	wiwiss.fu-berlin:interactionDrug1	wiwiss.fu-berlin:DB00008
wiwiss.fu-berlin:DB03585_DB01320	wiwiss.fu-berlin:interactionDrug1	wiwiss.fu-berlin:DB00008
wiwiss.fu-berlin:DB03585_DB01125	wiwiss.fu-berlin:interactionDrug1	wiwiss.fu-berlin:DB00008
wiwiss.fu-berlin:DB03585_DB00754	wiwiss.fu-berlin:interactionDrug1	wiwiss.fu-berlin:DB00008
wiwiss.fu-berlin:DB03585_DB00682	wiwiss.fu-berlin:interactionDrug1	wiwiss.fu-berlin:DB00008
wiwiss.fu-berlin:DB03585_DB01418	wiwiss.fu-berlin:interactionDrug2	wiwiss.fu-berlin:DB00008
wiwiss.fu-berlin:DB03585_DB01320	wiwiss.fu-berlin:interactionDrug2	wiwiss.fu-berlin:DB00008
wiwiss.fu-berlin:DB03585_DB01125	wiwiss.fu-berlin:interactionDrug2	wiwiss.fu-berlin:DB00008
wiwiss.fu-berlin:DB03585_DB00754	wiwiss.fu-berlin:interactionDrug2	wiwiss.fu-berlin:DB00008

Figure 6.1: Discovering synonymously used predicates with ProLod++

tion rule algorithm to the database management system. The loader provides an abstract layer, where data is transformed into the needed transactional databases according to the specified mining context and target. In the following, we further explain the mining-based features of ProLod++.

6.2 Mining-based Features

ProLod provided a simple association rule mining engine that allowed to discover positive and negative association rules among predicates [Böhm et al., 2010]. The algorithm used for mining is an adapted version of Aprioi [Agrawal and Srikant, 1994]. In ProLod++, we replace this system by a new framework adapting the mining configuration methodology that allows to mine each part of a statement as desired. The engine is able to discover positive association rules as well as negative association rules, so that we can implement the applications we introduced in the previous chapters.

6.2.1 Fact generation

ProLod++ provides two different methods to generate new facts based on a given dataset according to the algorithms described in Chapter 3. First, a user-driven approach supports the process of manual fact generation, and second, the

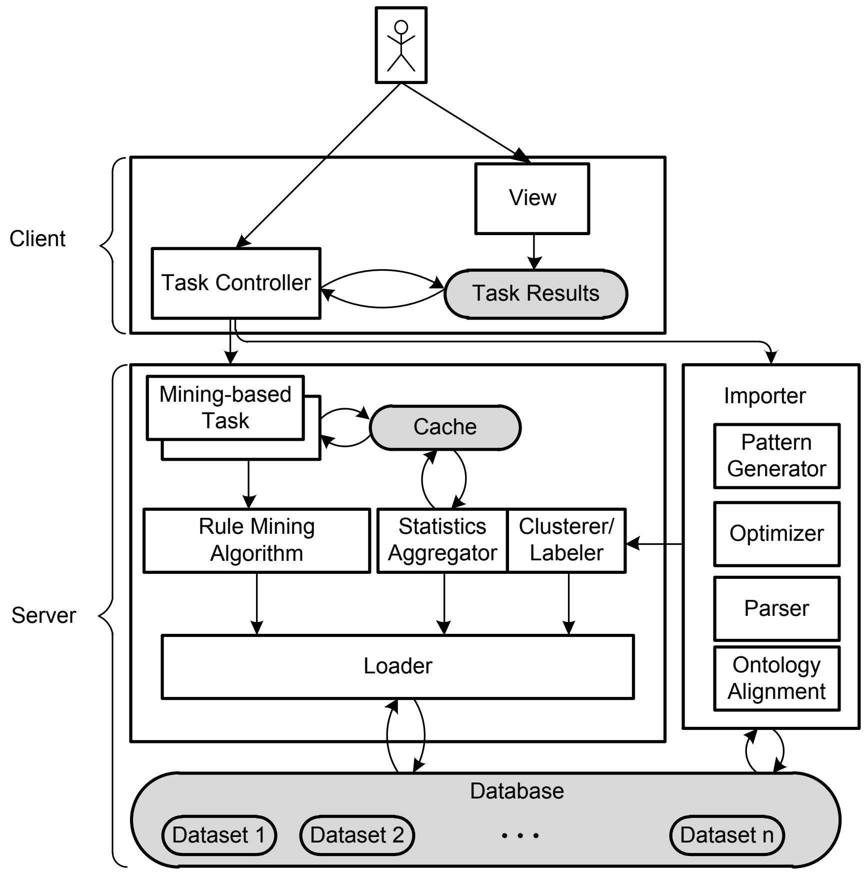


Figure 6.2: Component Overview of ProLod++

data-driven approach amends data with new triples.

Predicate and object suggestion

A user can add new facts to a given dataset via the ProLod++ GUI. It supports the user by suggesting predicate as well as object values according to association rules that have been mined from the dataset.

When a user is inserting or editing the facts related to a specific subject, the system is aware of all predicates or objects that have already been inserted for the current subject. To generate a list of suggestions, all rules that incorporate the previously inserted predicates as their antecedents are retrieved. The suggestions are all those predicates that occurred as consequences of the retrieved rules as described in Section 3.3. It is important to note that the user is required to choose subjects that already exist in the dataset. Furthermore, we currently do not materialize the new facts.

Auto-amendment of new facts

Following the data-driven approach presented in Section 3.4, ProLOD++ can automatically *amend* the given dataset with completely new facts, combining association rules between predicates and objects. By selecting one of the newly generated facts, ProLOD++ retrieves the corresponding high-confidence object rule.

6.2.2 Ontology change suggestions

ProLOD++ features our rule-based approach that automatically identifies over- and under-specifications and makes suggestions in order to change the ontology definition according to the descriptions in Chapter 4. With ProLOD++ the user can browse classes that are under- or over-specified. Furthermore, tables with suggestions to remove/add predicates from/to a certain class are presented; where, upon the selection of a suggested predicate, a user can identify the original domain class of the property.

The ontology change suggestion feature differs from all other features with regard to a technical point of view: As identifying over-and under-specifications of classes requires a time frame that is too high for live computation, we run the approach during the import phase.

6.2.3 Synonym analysis

Finally, ProLOD++ provides an interface to identify synonymously used predicates with the methodology presented in Chapter 5. Based on configurable thresholds, ProLOD++ generates synonymously used predicate candidates. For each candidate pair, ProLOD++ displays their joint frequency in the context of subjects and the schema score and ranks them according to the given schema score. We chose the reverse correlation coefficient (RCC) among the schema scores, because it is much more stable with regard to precision in most of the experiments, as we showed in Section 5.4 and is more comprehensive to users.

Again, by selecting a candidate pair, a sample set of triples will be presented, where the objects adhere to the range overlap of the predicates. This scenario is visualized in Figure 6.1, for the candidate pair `wiwiss.fu-berlin:interactionDrug1` and `wiwiss.fu-berlin:interactionDrug2` found in the Drugbank dataset[1].

[1]`http://datahub.io/dataset/fu-berlin-drugbank`

6.3 Summary

In this Chapter, we presented the co-developed tool ProLod++, with a special focus on the use cases and mining-based functionalities that have been described through out this thesis. ProLod++ is a proof-of-concept web-based system that enables users to brows and examine Rdf data in order to understand the data, identify patterns and outliers.

Currently, we restricted the usage of ProLod++ with regard to data import and re-clustering, due to hardware limitations, although the system already provides multi-user support. However, mining and profiling Rdf data is becoming more and more popular, and ever since ProLod was upgraded to ProLod++, the system receives more and more attention from the researchers of the Semantic Web community, who want to examine their data.

Chapter 7

Conclusions

Open RDF datasets, especially the widely deployed knowledge bases, suffer from inconsistency and incompleteness, which impede the usage and integration of those datasets for semantic applications. In this thesis, we presented approaches that improve the usability and quality of RDF datasets by means of association rule mining and frequency analysis. We examined association rule mining on statement level and introduced the methodology of mining configurations, which captures various possibilities to mine RDF data for different use cases. In particular, we applied our methodology to use cases that detect and can prevent inconsistency in RDF data.

By supporting *data creation*, we tackle both problems, the incompleteness of knowledge bases as well as the inconsistency in data. In Chapter 3, we showed how data creation can be supported via mining-based auto-completion. A fact about an entity, i.e., a fact with the entity URL as subject, can be auto-completed via suggestion of new predicates and objects. We identified that predicate suggestion usually is more reasonable than object suggestion and also leads to more promising results. By combining two mining configurations, we designed an automated approach for fact generation, which showed promising results on evolving datasets, such as DBpedia and YAGO.

In Chapter 4, we elaborated *schema analysis* and predicate mining in more detail. We identified misconceptions with regard to ontology usage and presented an algorithm to improve ontologies by aligning domains and range definitions of properties with the actual usage of the properties in the data. Our approach enables ontology designers to re-engineer the existing ontologies in order to eliminate ambiguity and redundancy in the vocabulary, and to prevent the creation of new proprietary properties.

In particular, we discovered that ontology discrepancies such as overspecification, occur because of the existence of synonymous predicates, which have been used in the data instead of the actual defined properties. In Chapter 5, we showed the importance of discovering such synonymously used predicates, which can in particular be used for expanding and relaxing SPARQL queries. Simulating schema-matching algorithms with our mining configurations, we presented an approach for automatically identifying synonymously used predicates in RDF data.

Finally, we presented our co-developed tool ProLOD++, which integrates adaptations of the above-mentioned approaches and serves as a proof-of-concept for profiling RDF datasets.

Based on the contributions of this thesis, there are several directions for further research that might be followed in the future:

Mining configurations. The methodology of mining configurations is not exhausted yet. Association rule mining at RDF statement level is an interesting field for further research, as there are remaining configurations, such as Configuration 3 and 5, to be elaborated and combined for other possible use cases. Beyond the various combination and refinement possibilities of configurations, the consideration of aggregated configurations by means of considering two parts instead of one part of a statement as either a mining context or target might also harbor interesting insights and applications. Finally, with regard to provenance analysis, one can easily extend the mining configuration methodology to quadruples, where the fourth element of a statement represents the *optional context value*, usually the provenance, of the triple. Considering the provenance field as a mining component in our mining configuration methodology might be relevant for the identification of provenance sources that have similar topics or structure. Those insights might help data consumers to better understand large and unknown datasets.

Combining the identified use-cases. Based on the mining configurations, we were able to generate new facts, to discover usage patterns of properties, and to identify synonymously used predicates in a dataset. In the evaluation section of Chapter 3, we stated that some of the generated facts are not included in later versions of a dataset because similar facts already exist, e.g., facts with predicates such as `country` or `nationality`. We believe that interlacing the synonym analysis with fact generation might lead to improved results with regard to property suggestion as well as fact generation.

Incorporating additional resources In this thesis, we limited the scope of data-driven solutions with regard to the data input, methodology and abstraction level in order to identify the capabilities of association rule mining and to keep our methodology as simple and general as possible:

- None of our approaches, which are designed to improve single RDF datasets, depends on external knowledge, such as text documents or dictionaries.
- Our frequency based analysis methods consider triple elements on graph level. In particular, we do not consider string similarities or data types.

Abolishing these restrictions for specific use cases are obvious directions for further research. Especially, combining external knowledge with frequency analysis on the actual dataset might contribute to the new use cases fact generation or fact ranking. We already started research on combining graph level knowledge with textual co-occurrences to assign global relevance scores for DBpedia facts [Langer et al., 2014], but interlacing the text co-occurrence scores with mining configurations is still an open field for future research.

Usability of open RDF data For the vision of the *web of data* to become reality, more efforts on improving the data quality need to be invested. The obvious dilemma of the web of data is the challenge that on the one hand, it should grow as fast as possible but on the other hand, the faster it grows the harder it becomes to cope with quality issues. To handle this tradeoff, a balanced mix of manual curation and automatic approaches are needed. While automated approaches for extracting data contribute to the rapid growth of data, the accuracy and correctness of the data suffer to some extent, because today the extraction algorithms still cannot capture *the semantics of a text*. Therefore, it is necessary to supervise and curate generated data on regular basis. Clearly, with the growth of data, its manual curation gets more and more cumbersome. Precisely for that reason, it is important to have analysis tools and methods that identify snapshots of existing patterns and problems in the data. If such a process is systematically repeated, analysis tools also improve on their accuracy, because any algorithm for pattern and outlier analysis requires some degree of consistency in a dataset. If you put clean data in, clean results will come out. Improving the machine-readability of existing data leads to extraction algorithms with better look-up options and opportunities to *understand semantics*, which in turn leads to better quality of the algorithm's output.

Bibliography

[Abedjan and Naumann, 2011] Z. Abedjan and F. Naumann. Context and target configurations for mining RDF data (2 pages). In *Proceedings of the International Workshop on Search and Mining Entity-Relationship Data (SMER)*, pages 23–24, Glasgow, 2011.

[Abedjan and Naumann, 2013a] Z. Abedjan and F. Naumann. Improving rdf data through association rule mining. *Datenbank-Spektrum*, 13(2):111–120, 2013a.

[Abedjan and Naumann, 2013b] Z. Abedjan and F. Naumann. Synonym analysis for predicate expansion. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 140–154, Montpellier, France, 2013b.

[Abedjan et al., 2012] Z. Abedjan, J. Lorey, and F. Naumann. Reconciling ontologies and the web of data. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1532–1536, New York, NY, USA, 2012.

[Abedjan et al., 2014] Z. Abedjan, T. Gruetze, A. Jentzsch, and F. Naumann. Profiling and mining rdf data with prolod++. In *Proceedings of the International Conference on Data Engineering (ICDE)*, Chicago, IL, 0 2014.

[Agrawal and Srikant, 1994] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 487–499, Santiago de Chile, Chile, 1994.

[Agrawal et al., 1993] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 207–216, Washington, D.C., USA, 1993.

[Aguirre et al., 2012] J.-L. Aguirre, K. Eckert, J. Euzenat, A. Ferrara, W. R. van Hage, L. Hollink, C. Meilicke, A. Nikolov, D. Ritze, F. Scharffe, P. Shvaiko, O. Sváb-

Zamazal, C. T. dos Santos, E. Jiménez-Ruiz, B. C. Grau, and B. Zapilko. Results of the ontology alignment evaluation initiative 2012. In *OM*, 2012.

[Antonie and Zaïane, 2004] M.-L. Antonie and O. R. Zaïane. Mining positive and negative association rules: an approach for confined rules. In *European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 27–38, New York, NY, USA, 2004.

[Auer et al., 2012] S. Auer, J. Demter, M. Martin, and J. Lehmann. Lodstats – an extensible framework for high-performance dataset analytics. In *EKAW*, volume 7603, pages 353–362, 2012. ISBN 978-3-642-33875-5.

[Baader et al., 2005] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In D. Hutter and W. Stephan, editors, *Mechanizing Mathematical Reasoning*, volume 2605 of *Lecture Notes in Computer Science*, pages 228–248. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25051-7.

[Baeza-Yates and Ribeiro-Neto, 1999] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[Baroni and Bisi, 2004] M. Baroni and S. Bisi. Using cooccurrence statistics and the web to discover synonyms in technical language. In *International Conference on Language Resources and Evaluation*, pages 1725–1728, 2004.

[Bilke and Naumann, 2005] A. Bilke and F. Naumann. Schema matching using duplicates. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 69–80, April 2005.

[Bizer et al., 2009a] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5 (3):1–22, 2009a.

[Bizer et al., 2009b] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A crystallization point for the Web of Data. *Journal of Web Semantics (JWS)*, 7:154–165, September 2009b.

[Böhm et al., 2010] C. Böhm, F. Naumann, Z. Abedjan, D. Fenz, T. Grütze, D. Hefenbrock, M. Pohl, and D. Sonnabend. Profiling linked open data with ProLOD. In *Proceedings of the International Workshop on New Trends in Information Integration (NTII)*, pages 175–178, 2010.

[Böhm et al., 2012] C. Böhm, G. de Melo, F. Naumann, and G. Weikum. Linda: distributed web-of-data-scale entity matching. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 2104–2108, 2012.

[Bonatti et al., 2011] P. A. Bonatti, A. Hogan, A. Polleres, and L. Sauro. Robust and scalable linked data reasoning incorporating provenance and trust annotations. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2), 2011. ISSN 1570-8268.

[Brin et al., 1997] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 255–264, New York, NY, USA, 1997.

[Buitelaar and Cimiano, 2008] P. Buitelaar and P. Cimiano, editors. *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, volume 167 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, 2008.

[Cafarella et al., 2008] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1:538–549, August 2008.

[Cheng et al., 2011a] G. Cheng, T. Tran, and Y. Qu. Relin: relatedness and informativeness-based centrality for entity summarization. In *Proceedings of the International Conference on Semantic Web (ISWC)*, pages 114–129, 2011a.

[Cheng et al., 2011b] G. Cheng, T. Tran, and Y. Qu. Relin: relatedness and informativeness-based centrality for entity summarization. In *Proceedings of the International Conference on Semantic Web (ISWC)*, pages 114–129, 2011b.

[Chi et al., 2004] Y. Chi, R. R. Muntz, S. Nijssen, and J. N. Kok. Frequent subtree mining – an overview. *Fundamenta Informaticae*, 66:161–198, November 2004. ISSN 0169-2968.

[d'Amato et al., 2008] C. d'Amato, N. Fanizzi, and F. Esposito. Query answering and ontology population: An inductive approach. In *ESWC*, pages 288–302, 2008.

[d'Amato et al., 2010] C. d'Amato, N. Fanizzi, and F. Esposito. Inductive learning for the semantic web: What does it buy? *Semantic Web Journal*, 1(1,2):53–59, Apr. 2010. ISSN 1570-0844.

[d'Amato et al., 2012] C. d'Amato, V. Bryl, and L. Serafini. Semantic knowledge discovery from heterogeneous data sources. In *Knowledge Engineering and Knowledge Management (EKAW)*, pages 26–31, 2012.

[Das et al., 2001] A. Das, W.-K. Ng, and Y.-K. Woon. Rapid association rule mining. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 474–481, New York, NY, USA, 2001.

[Davis and Goadrich, 2006] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 233–240, New York, NY, USA, 2006.

[Dehaspe and Toivonen, 1999] L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining anf Knowledge Discovery*, 3(1):7–36, Mar. 1999.

[Deshpande and Karypis, 2004] M. Deshpande and G. Karypis. Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.*, 22:143–177, January 2004.

[Ding et al., 2005] L. Ding, L. Zhou, T. Finin, and A. Joshi. How the semantic web is being used: An analysis of foaf documents. In *Proceedings of the Annual Hawaii International Conference on System Sciences (HICSS)*, page 113.3, Big Island, HI, 2005.

[Ding et al., 2010] L. Ding, J. Shinavier, Z. Shangguan, and D. McGuinness. Sameas networks and beyond: Analyzing deployment status and implications of owl:sameas in linked data. In P. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Pan, I. Horrocks, and B. Glimm, editors, *The Semantic Web ,Äì ISWC 2010*, volume 6496 of *Lecture Notes in Computer Science*, pages 145–160. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-17745-3.

[Doan et al., 2001] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 509–520, New York, NY, 2001.

[Elbassuoni et al., 2011] S. Elbassuoni, M. Ramanath, and G. Weikum. Query relaxation for entity-relationship search. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 62–76, Berlin, Heidelberg, 2011. Springer-Verlag.

[Elbassuoni et al., 2012] S. Elbassuoni, M. Ramanath, and G. Weikum. RDF Xpress: a flexible expressive RDF search engine. pages 1013–1013, New York, NY, USA, 2012.

[Elmagarmid et al., 2007] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(1):1–16, 2007.

[Fanizzi et al., 2008] N. Fanizzi, C. d'Amato, and F. Esposito. Conceptual clustering and its application to concept drift and novelty detection. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 318–332, 2008.

[Fanizzi et al., 2009] N. Fanizzi, C. d'Amato, and F. Esposito. Inductive classification of semantically annotated resources through reduced coulomb energy networks. *International Journal on Semantic Web and Information Systems*, 5(4):19–38, 2009.

[Fleischhacker et al., 2012] D. Fleischhacker, J. Völker, and H. Stuckenschmidt. Mining rdf data for property axioms. In *On the Move to Meaningful Internet Systems: OTM 2012*, volume 7566 of *Lecture Notes in Computer Science*, pages 718–735. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33614-0.

[Galàrraga et al., 2013] L. Galàrraga, C. Teflioudi, K. Hose, and F. Suchanek. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the International World Wide Web Conference (WWW)*, Rio, Brasil, 2013.

[Golbeck and Rothstein, 2008] J. Golbeck and M. Rothstein. Linking Social Networks on the Web with FOAF: A Semantic Web Case Study. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1138–1143, Chicago, IL, 2008.

[Gottlob and Senellart, 2010] G. Gottlob and P. Senellart. Schema mapping discovery from data instances. *Journal of the ACM*, 57(2):6:1–6:37, 2010. ISSN 0004-5411.

[Grigonytė et al., 2010] G. Grigonytė, J. a. Cordeiro, G. Dias, R. Moraliyski, and P. Brazdil. Paraphrase alignment for synonym evidence discovery. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, COLING '10, pages 403–411, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[Guarino, 1998] N. Guarino. *Formal Ontology in Information Systems.* IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st edition, 1998. ISBN 9051993994.

[Guarino and Welty, 2002] N. Guarino and C. Welty. Evaluating ontological decisions with ontoclean. *Commun. ACM*, 45(2):61–65, Feb. 2002. ISSN 0001-0782.

[Haase et al., 2005] P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, and Y. Sure. A framework for handling inconsistency in changing ontologies. In *Proceedings of the International Conference on Semantic Web (ISWC)*, pages 353–367, Berlin, Heidelberg, 2005. Springer-Verlag.

[Han et al., 2000] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 1–12, 2000. URL `citeseer.nj.nec.com/han99mining.html`.

[Harris, 1954] Z. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[Hayes-Roth et al., 1983] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat. *Building Expert Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1983. ISBN 0-201-10686-8.

[Heath and Bizer, 2011] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 1st edition, 2011. URL `http://linkeddatabook.com/`.

[Heim et al., 2010] P. Heim, S. Lohmann, and T. Stegemann. Interactive relationship discovery via the semantic web. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, volume 6088, pages 303–317, 2010.

[Hoffart et al., 2013] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, Jan. 2013. ISSN 0004-3702.

[Hogan et al., 2009] A. Hogan, A. Harth, and A. Polleres. Scalable authoritative owl reasoning for the web. *Int. J. Semantic Web Inf. Syst.*, 5(2):49–90, 2009.

[Hogan et al., 2012] A. Hogan, J. Umbrich, A. Harth, R. Cyganiak, A. Polleres, and S. Decker. An empirical survey of linked data conformance. *Web Semant.*, 14: 14–44, July 2012. ISSN 1570-8268.

[Horrocks et al., 2003] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From shiq and rdf to owl: the making of a web ontology language. *J. Web Sem.*, 1 (1):7–26, 2003.

[Isele and Bizer, 2012] R. Isele and C. Bizer. Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, 5(11):1638–1649, July 2012. ISSN 2150-8097.

[Isele et al., 2011] R. Isele, A. Jentzsch, and C. Bizer. Efficient multidimensional blocking for link discovery without losing recall. In *Proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB)*, 2011.

[Joshi et al., 2013] A. K. Joshi, P. Hitzler, and G. Dong. Logical linked data compression. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 170–184, 2013.

[Józefowska et al., 2010] J. Józefowska, A. Lawrynowicz, and T. Lukaszewski. The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *Theory Pract. Log. Program.*, 10:251–289, 2010. ISSN 1471-0684.

[Kandel et al., 2012] S. Kandel, R. Parikh, A. Paepcke, J. Hellerstein, and J. Heer. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Advanced Visual Interfaces*, 2012.

[Kasneci et al., 2009a] G. Kasneci, S. Elbassuoni, and G. Weikum. Ming: mining informative entity relationship subgraphs. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1653–1656, 2009a.

[Kasneci et al., 2009b] G. Kasneci, S. Elbassuoni, and G. Weikum. Ming: mining informative entity relationship subgraphs. In *CIKM*, pages 1653–1656, 2009b.

[Kasneci et al., 2009c] G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum. The yago-naga approach to knowledge discovery. *SIGMOD Record*, 37(4):41–47, 2009c.

[Kinsella et al., 2008] S. Kinsella, U. Bojars, A. Harth, J. G. Breslin, and S. Decker. An interactive map of semantic web ontology usage. In *Proceedings of the International Conference on Information Visualisation (IV)*, pages 179–184, Washington, DC, 2008.

[Knuth et al., 2012] M. Knuth, J. Hercher, and H. Sack. Collaboratively patching linked data. In *International Workshop on Usage Analysis and the Web of Data (USEWOD)*, volume abs/1204.2715, 2012.

[Koudas et al., 2006] N. Koudas, C. Li, A. K. H. Tung, and R. Vernica. Relaxing join and selection queries. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 199–210. VLDB Endowment, 2006.

[Kuramochi and Karypis, 2001] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 313–320, Washington, D.C., 2001.

[Lange et al., 2010] D. Lange, C. Böhm, and F. Naumann. Extracting structured information from Wikipedia articles to populate infoboxes. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1661–1664, New York, NY, USA, 2010. ISBN 978-1-4503-0099-5.

[Langer et al., 2014] P. Langer, P. Schulze, S. George, M. Kohnen, T. Metzke, Z. Abedjan, and G. Kasneci. Assigning global relevance scores to dbpedia facts. In *International Workshop on Data Engineering meets the Semantic Web (DESWeb)*, Chicago, IL, 0 2014. URL `file:55394--pdf`.

[Lehmann and Bühmann, 2010] J. Lehmann and L. Bühmann. Ore - a tool for repairing and enriching knowledge bases. In *Proceedings of the International Conference on Semantic Web (ISWC)*, pages 177–193, Berlin, Heidelberg, 2010. Springer-Verlag.

[Li et al., 2009] J. Li, J. Tang, Y. Li, and Q. Luo. Rimom: A dynamic multistrategy ontology alignment framework. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 21(8):1218–1232, 2009. ISSN 1041-4347.

[Li and Clifton, 2000] W.-S. Li and C. Clifton. Semint: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data and Knowledge Engineering (DKE)*, 33(1):49 – 84, 2000. ISSN 0169-023X.

[Lisi and Esposito, 2005] F. A. Lisi and F. Esposito. Mining the semantic web: A logic-based methodology. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems (ISMIS)*, pages 102–111, Saratoga Springs, 2005.

[Lorey et al.] J. Lorey, Z. Abedjan, F. Naumann, and C. Böhm. RDF Ontology (Re-)Engineering through Large-scale Data Mining. Contribution to the Billion Triple Challenge at International Semantic Web Conference (ISWC) 2011.

[Maedche and Staab, 2001] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16:72–79, March 2001. ISSN 1541-1672.

[Menezes et al., 2010] G. V. Menezes, J. M. Almeida, F. Belém, M. A. Gonçalves, A. Lacerda, E. S. De Moura, G. L. Pappa, A. Veloso, and N. Ziviani. Demand-driven tag recommendation. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pages 402–417, 2010.

[Muggleton, 1995] S. Muggleton. Inverse Entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995. URL `citeseer.nj.nec.com/muggleton95inverse.html`.

[Mutharaju, 2012] R. Mutharaju. Very large scale owl reasoning through distributed computation. In *Proceedings of the International Conference on Semantic Web (ISWC)*, pages 407–414, Berlin, Heidelberg, 2012. Springer-Verlag.

[Naumann, 2013] F. Naumann. Data profiling revisited. *SIGMOD Record*, 2013. to appear.

[Naumann et al., 2002] F. Naumann, C.-T. Ho, X. Tian, L. M. Haas, and N. Megiddo. Attribute classification using feature analysis. In *Proceedings of the International Conference on Data Engineering (ICDE)*, page 271, 2002.

[Nebot and Berlanga, 2010] V. Nebot and R. Berlanga. 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)*, volume 2, pages 504–513, Cordoba, Spain, 2010. ISBN 3-642-13024-0, 978-3-642-13024-3.

[Page et al., 1998] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *WWW Internet And Web Information Systems*, 54(2):1–17, 1998.

[Park et al., 1997] J. S. Park, M.-S. Chen, and P. S. Yu. Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 9:813–825, September 1997. ISSN 1041-4347.

[Parundekar et al., 2010] R. Parundekar, C. A. Knoblock, and J. L. Ambite. Linking and building ontologies of linked data. In *Proceedings of the International Conference on Semantic Web (ISWC)*, pages 598–614, Berlin, Heidelberg, 2010. Springer-Verlag.

[Rahm and Bernstein, 2001] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, Dec. 2001. ISSN 1066-8888.

[Rettinger et al., 2009] A. Rettinger, M. Nickles, and V. Tresp. Statistical relational learning with formal ontologies. In *European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 286–301, 2009.

[Rettinger et al., 2012] A. Rettinger, U. Lösch, V. Tresp, C. d'Amato, and N. Fanizzi. Mining the semantic web - statistical learning for next generation knowledge bases. *Data Min. Knowl. Discov.*, 24(3):613–662, 2012.

[Schoenmackers et al., 2010] S. Schoenmackers, O. Etzioni, D. S. Weld, and J. Davis. Learning first-order horn clauses from web text. pages 1088–1098, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[Stoilos et al., 2005] G. Stoilos, G. Stamou, V. Tzouvaras, J. Z. Pan, and I. Horrocks. The fuzzy description logic f-shin. In *Proceedings of the International Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, 2005.

[Stoilos et al., 2007] G. Stoilos, G. Stamou, J. Z. Pan, V. Tzouvaras, and I. Horrocks. Reasoning with very expressive fuzzy description logics. *J. Artif. Intel. Res.*, 30(1):273–320, Oct. 2007. ISSN 1076-9757.

[Stojanovic et al., 2003] L. Stojanovic, N. Stojanovic, J. Gonzalez, and R. Studer. Ontomanager ‚Äì a system for the usage-based ontology management. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 858–875. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-20498-5.

[Stojanovic and Stojanovic, 2002] N. Stojanovic and L. Stojanovic. Usage-oriented evolution of ontology-based knowledge management systems. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, volume 2519 of *Lecture Notes in Computer Science*, pages 1186–1204. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-00106-5. doi: 10.1007/3-540-36124-3_75.

[Stonebraker et al., 2013] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu. Data curation at scale: The data tamer system. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2013.

[Stumme et al., 2006] G. Stumme, A. Hotho, and B. Berendt. Semantic web mining. *Journal of Web Semantics*, 4(2):124–143, June 2006. ISSN 1570-8268.

[Suchanek et al., 2007] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 697–706, 2007.

[Suchanek et al., 2011] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *Proceedings of the VLDB Endowment*, 5(3):157–168, 2011.

[Suchanek et al., 2013] F. M. Suchanek, J. Hoffart, E. Kuzey, and E. Lewis-Kelham. Yago2s: Modular high-quality information extraction with an application to flight planning. In *Proceedings of the Conference Datenbanksysteme in Büro, Technik und Wissenschaft (BTW)*, pages 515–518, 2013.

[Sydow et al., 2013] M. Sydow, M. Pikula, and R. Schenkel. The notion of diversity in graphical entity summarisation on semantic knowledge graphs. *J. Intell. Inf. Syst.*, 41(2):109–149, 2013.

[Thor et al., 2011] A. Thor, P. Anderson, L. Raschid, S. Navlakha, B. Saha, S. Khuller, and X.-N. Zhang. Link prediction for annotation graphs using graph summarization. In *Proceedings of the International Conference on Semantic Web (ISWC)*, pages 714–729, 2011.

[Töpper et al., 2012] G. Töpper, M. Knuth, and H. Sack. Dbpedia ontology enrichment for inconsistency detection. In *Proceedings of the International Conference on Semantic Systems (I-Semantics)*, pages 33–40, New York, NY, USA, 2012.

[Turney, 2001] P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the European Conference on Machine Learning (EMCL)*, pages 491–502, London, UK, UK, 2001. Springer-Verlag.

[Udrea et al., 2007] O. Udrea, L. Getoor, and R. J. Miller. Leveraging data and structure in ontology integration. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 449–460, New York, NY, USA, 2007.

[Urbani et al., 2010] J. Urbani, S. Kotoulas, J. Maassen, F. van Harmelen, and H. Bal. Owl reasoning with webpie: Calculating the closure of 100 billion triples. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 213–227, Berlin, Heidelberg, 2010. Springer-Verlag.

[Völker and Niepert, 2011] J. Völker and M. Niepert. Statistical schema induction. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 124–138, Heraklion, Greece, 2011.

[Volz et al., 2009] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk - a link discovery framework for the web of data. In *LDOW*, 2009.

[Wei et al., 2009] X. Wei, F. Peng, H. Tseng, Y. Lu, and B. Dumoulin. Context sensitive synonym discovery for web search queries. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1585–1588, New York, NY, USA, 2009.

[Wu and Weld, 2007] F. Wu and D. S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 41–50, New York, NY, USA, 2007.

[Wu and Weld, 2008] F. Wu and D. S. Weld. Automatically refining the Wikipedia infobox ontology. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 635–644, Beijing, China, 2008.

[Wu et al., 2004] X. Wu, C. Zhang, and S. Zhang. Efficient mining of both positive and negative association rules. *ACM Transactions on Information Systems*, 22 (3):381–405, July 2004. ISSN 1046-8188.

[Zaki, 2000] M. J. Zaki. Scalable Algorithms for Association Mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 12:372–390, May 2000. ISSN 1041-4347.

[Zhou et al., 2007] X. Zhou, J. Gaugaz, W.-T. Balke, and W. Nejdl. Query relaxation using malleable schemas. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 545–556, New York, NY, USA, 2007.

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Doktorarbeit mit dem Thema:
Improving RDF Data with Data Mining
selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Potsdam, den 3. März 2014

Ziawasch Abedjan