AUTOMATED KNOWLEDGE ENRICHMENT FOR SEMANTIC WEB DATA

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Supervisors:

Dr. Quan Bai

Dr. Jian Yu

Dr. Qing Liu

November 2018

By

Molood Barati

School of Engineering, Computer and Mathematical Sciences

Copyright

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

Signature of candidate

Acknowledgements

First and foremost, I would like to express my special gratitude to my primary supervisor Dr. Quan Bai, who has been a professional advisor for me. His ethics, sincerity, friendship, and motivation have profoundly inspired me. He has taught me to present my ideas as clearly and simply as possible. Apart from academic achievements, he gifted me the most important lesson of my life that is "patience". Without patience, nothing great can be achieved.

I would also like to acknowledge my mentor supervisor, Dr. Qing Liu, for her constructive advice. The completion of this thesis would not have been possible without her scientific supports and suggestions. Special thanks to my colleagues, within the School of Engineering, Computer and Mathematical Sciences at Auckland University of Technology, who I have had the pleasure to work with them during this time, Dr. Jian Yu, Prof, Ajit Narayanan, Helena Bahrami, etc.

Words can not express how grateful I am to my family in the pursuit of this project. I would like to appreciate my parents and sister, whose endless love supported me during this journey. Special thanks are due to my beloved husband, Jamal, for his continuous assist and understanding in the completion of this thesis.

Abstract

The Semantic Web is an effort to interchange unstructured data over the Web into a structured format that is processable not only by human beings but also computers. The Semantic Web creates a distributed framework to publish, query, and reuse information. The key backbones of Semantic Web are ontologies and annotations that provide semantics for raw data known as RDF data.

Although there exist many Semantic Web applications, sophisticated analytical infrastructures are still lacking, preventing users from extracting the semantics attached to RDF data. Additionally, the Semantic Web data face with a wide range of data quality issues due to the distributed nature of the Semantic Web. This thesis presents three approaches based on the following purposes: (I) to express the semantics behind discovered patterns, (II) to deal with a Semantic Web data quality issue, and (III) to enrich knowledge in the Semantic Web ontologies.

The following contributions have been made in this thesis. Firstly, the thesis shows the influence of relations and ontological knowledge in the process of mining hidden patterns and proposes Semantic Web Association Rule Mining (SWARM), an automated mining approach that attaches semantics to the discovered patterns. Secondly, the thesis concentrates on a data quality issue in the Semantic Web field which indicates incorrect assignments between instances and classes in the ontology. To this end, Class Assignment Detector (CAD) approach has been designed to tackle the data quality issue. Thirdly, the thesis enhances the process of ontology enrichment by generating new classes by mining instance-level and schema-level knowledge. Since ontologies are often designed before actual usage, Class Enricher (CEn) approach is developed to extract new classes which are not defined in the ontologies. All the proposed approaches have been tested over real datasets to validate their effectiveness.

Publications

1. Barati, M., Bai, Q. & Liu, Q. (2016). Swarm: an approach for mining semantic association rules from semantic web data. *In Pacific rim international conference on artificial intelligence* (pp. 30–43). Springer.

2. Barati, M., Bai, Q. & Liu, Q. (2017). Mining semantic association rules from rdf data. *Knowledge-Based Systems, 133*, 183–196. Elsevier.

3. Barati, M., Bai, Q. & Liu, Q. (2018). An entropy-based class assignment detection approach for rdf data. *In Pacific rim international conference on artificial intelligence* (pp. 412–420). Springer.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Semantic Web (SW) promised the possibility of a machine and human understandable Web environment (Berners-Lee, Hendler & Lassila, 2001; Bizer, Heath & Berners-Lee, 2011). The SW aims to convert the current Web (i.e., Web 2.0) data expressed by unstructured and semi-structured formats into a Web of data. In this regards, the SW data should be accessible via a standard format. Relationships among data should also be available to connect resources from different datasets. To this end, the SW data are usually structured in the triple format (subject, predicate, object) called Resource Description Framework (RDF). In an RDF triple, a subject and an object are resources and literals, while a predicate indicates a meaningful relationship between the subject and object. The RDF data can also be published over Linked Open Data (LOD) cloud in various formats such as RDFa, N-Triples, and Turtle.

In recent years, the number of large Knowledge Bases (KBs) available on the LOD cloud is noticeably growing through developing the SW technologies such as RDF Schema (RDF/S) and Web Ontology Language (OWL) standardization. One of the main features of the SW is to provide a decentralized platform so that different KBs can easily link to each other. In this respect, RDF-style KBs such as DBpedia (Auer et al., 2007) and YAGO (Suchanek, Kasneci & Weikum, 2007) have a high contribution of distributing SW data over the LOC cloud. These KBs contain two main levels, i.e., instance-level that contains raw RDF triples and schema-level (ontology) that provides semantics for the triples.

The SW technologies enable data providers to create and publish their own RDF-style KBs. Large KBs contain hundreds of millions of RDF triples that are managed by the SW communities. Consequently, mining and learning from the SW data are very challenging due to several reasons. Generally, these KBs usually contain a certain level of noisy data because of the distributed nature of the SW. Besides that, the existing SW data mining approaches do not usually take advantage of the semantic potential provided by the schema-level knowledge.

This thesis presents mining and learning approaches for three main purposes: (I) to show the importance of using relations and schema-level knowledge in the mining process, (II) to tackle a SW data quality issue that indicates incorrect assignments between instances and classes in the ontology, and (III) to enrich schema-level knowledge by discovering new classes. In this chapter, Section 1.1 briefly reviews the main concepts in the context of the SW. Section 1.2 discusses major challenges in the SW. The motivations used throughout the thesis are explained in Section 1.3. Section 1.4 explains the research questions and objectives of this study. The contributions, research methodology, and thesis structure are given in sections 1.5-1.7, respectively.

1.1 Preliminaries

The SW technologies aim to represent data in a machine-understandable format to integrate data from different resources. These technologies allow users to store data which are not restricted by a particular schema (Shadbolt, Berners-Lee & Hall, 2006). Figure 1.1 shows the SW stack, known as "layer cake" proposed by Tim Berners-Lee. In the SW, every entity has a specific name identified by one Uniform Resource Identifier

(URI). The eXtensible Markup Language (XML) is a format enables users to publish structured Web documents. As shown in the figure, the RDF layer is placed on top of the XML layer since RDF data can be expressed in a directed-labeled graph, while an XML tree is an unlabelled and undirected graph. The RDF data model allows structured and semi-structured data to be merged and distributed among various applications. The RDF/S adds more semantics to RDF data by providing basic vocabularies. To enrich RDF/S and describe more complex RDF statements, developers came up with the idea of a powerful ontology language called OWL. The logic layer improves OWL by providing more specific declarative knowledge that allows users to create logical relations that can not be described by OWL. On top of the Logic layer, there exists another layer called proof layer that is designed for validating RDF statements. Finally, Trust layer is created to evaluate digital signatures (Horrocks & Patel-Schneider, 2003; Antoniou & Van Harmelen, 2004).

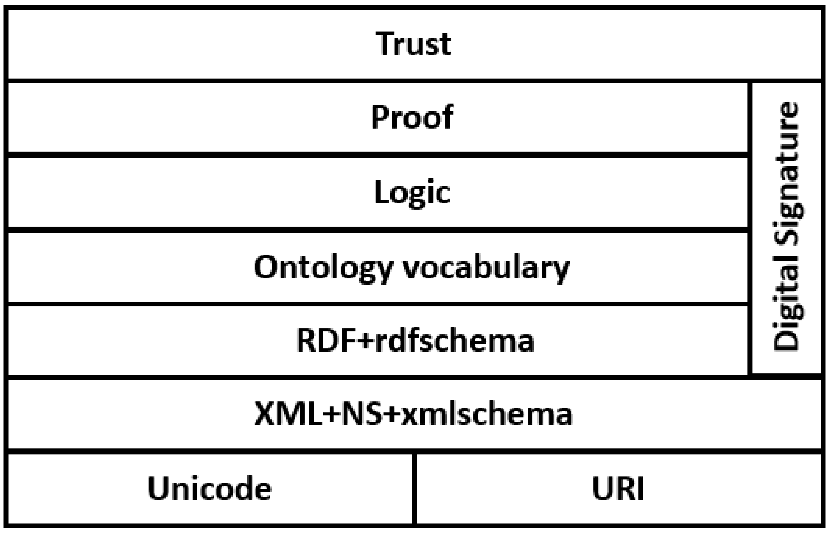

Figure 1.1: The SW layer cake (Horrocks & Patel-Schneider, 2003)

1.1.1 RDF data model

The RDF data model originating from the SW is a knowledge representation method that enables free data exchange and formalisation of knowledge. The RDF is known as a standard way to describe conceptual information available on the Web. The RDF data is usually structured in a triple format, i.e., (subject, predicate, object), that can be modelled as a directed-labeled graph. The model can be easily implemented and navigated by its simple representation. The RDF data model also facilitates the process of data merging from various KBs (Consortium et al., 2014).

1.1.2 RDF/S

RDF/S is a collection of classes with specific properties to enrich RDF triples by adding semantics. The primary goal of RDF/S is to be used as a language to build ontological structures by defining classes and properties. RDF/S presents classes along with domain and range restrictions for properties. Additionally, it shows the hierarchies of classes and properties by subsumption relationships (Decker et al., 2000; Broekstra, Kampman & Van Harmelen, 2003).

1.1.3 OWL

OWL is a SW language used for creating and distributing ontologies over the LOD cloud (Bao et al., 2009; Schneider, Carroll, Herman & Patel-Schneider, 2009). OWL provides more rich and complex relationships for RDF triples as compared to RDF/S language. In the context of the SW, ontologies are the foundations that build a network of information with logical relations (Motik, Sattler & Studer, 2005; Goczyła, Grabowska, Waloszek & Zawadzki, 2006; Alexander & Hausenblas, 2009).

1.1.4 SPARQL

The SW requires a semantic query language to retrieve RDF data from divers RDF-style KBs. SPARQL Protocol and RDF Query Language (SPARQL) is a standardized graph-based language for querying and retrieving graph patterns in the SW. The SPARQL facilitates different operations such as aggregation, negation, creating values using expressions, sub-queries, and extensible value testing (Quilitz & Leser, 2008). It also provides some capabilities such as editing, adding, and removing RDF triples as well as updating, creating, and removing RDF graphs (Saleem, Ali, Hogan, Mehmood & Ngomo, 2015).

1.1.5 LOD

The LOD is a method proposed by Tim Berners-Lee for publishing RDF-style KBs. The LOD mainly concentrates on distributing structured data that are usually represented in the RDF data format. The LOD method builds upon World Wide Web (WWW) technologies, and it facilitates data from diverse resources to be linked and queried (Freitas, Curry, Oliveira & O'Riain, 2012; Acosta et al., 2013; Schmachtenberg, Bizer & Paulheim, 2014). The LOD represents a linked data graph among different KBs. The tendency of publishing RDF data is to keep growing, and data providers estimated that there exist about 1,220[1] RDF-style KBs over the LOD cloud by June 2018.

Tim Berners-Lee defines four principles for sharing KBs over the LOD: (I) Data providers should define Internationalized Resource Identifiers (IRIs) for identifying things, (II) Data providers should use IRIs in the standard protocols, so that these things can be referred to and dereferenced by users, (III) Data providers should publish data in a standard format such as RDF data model, and (IV) Data providers are strongly recommended to use links among sources for enhancing the process of data discovery.

[1]https://lod-cloud.net/

1.2 Challenging research issues in the SW

The SW technologies allow users to create and distribute their own KBs. However, the released KBs are very diverse in terms of formats, structure, and vocabulary. Such diversities cause different issues that are briefly explained as follows.

- **Integration and Interoperability**: In the SW, ontologies facilitate the process of knowledge sharing and reuse. Currently, the number of various ontological structures that have modelled similar domains of knowledge is quickly growing. On this subject, an important issue is to develop a standardized mapping mechanism to create a more unified ontological structure for sharing knowledge. Since ontologies evolve over time, extending and updating existing ontologies is another important issue in the SW field.

- **Query search mechanism**: RDF data can be represented as a directed-labeled graph. SPARQL is a graph-matching query language designed for the SW data. Although SPARQL facilitates the use of the SW data, it is not intelligent enough for discovering complex patterns. To overcome this limitation, researchers usually borrow data mining techniques to analyse the SW data. Accordingly, enhancing the performance of SPARQL for discovering the most relevant answers is a hot topic in the SW field.

- **SW data mining techniques**: Recently, researchers are targeting two demanding research areas together: SW and data mining. The primary purpose of this line is to extract hidden patterns from the SW by utilising data mining techniques. The SW is a Web of data that describe relations (i.e., predicates) among resources. A major difference of the SW with the Web 2.0 is that the SW relations are named to express meaningful interactions among resources. Furthermore, the SW data are conceptually enriched by knowledge at the schema-level. The effect of such potentials on the semantics of discovered patterns is not explicitly studied by most existing SW data mining approaches.

- **SW data quality**: Ontologies are the backbone of the SW that are usually created independently before actual data is populated. Subsequently, ontologies can be incomplete and they often do not provide all aspects that are required for specific domains of knowledge. Additionally, publishing the SW data over the LOD cloud is maintained by a community of thousands of contributors. It is a very time-consuming and error-prone process due to a large amount of data. Therefore, the SW data suffer from various forms of quality issues.

It is important to mention that this thesis concentrates on some specific shortcomings in the SW data mining techniques and the SW data quality which are described in the following section.

1.3 Motivation

The following explains how particular issues in the SW data mining techniques and the SW data quality motivated us to the topic of this thesis.

- **Lack of semantic extraction:** In the context of the SW data mining, different approaches have been proposed to discover frequent patterns from the SW data. Most existing approaches usually consider knowledge at the instance-level. More precisely, the potential of schema-level knowledge is disregarded by these approaches. Ignoring knowledge at the schema-level can have negative impacts on interpreting the patterns. For example, consider *Thomas Aquinas* ⇒ *Plato*, *Duns Scotus* as a frequent pattern discovered by mining RDF triples. In DBpedia ontology, *Thomas Aquinas* belongs to Saint class, while *Plato* and *Duns Scotus* have been assigned to Philosopher class. Now the question is that whether the pattern express a group of saints or philosophers? In order to interpret the above pattern, if we consider *Thomas Aquinas*, *Plato* and *Duns Scotus* as members of a general class such as the Person class, we might reduce the semantics of pattern since upper classes in the ontology share more general descriptions to compare with lower classes that express more specific descriptions. Although, the pattern shows an association between *Thomas Aquinas*, *Plato*, and *Duns Scotus*, the relations (predicates) among entities (i.e., subjects and objects) are not explicitly described. Based on these motivations, Chapter 3 of this thesis introduces an approach that shows how to utilise instance-level and schema-level knowledge to generate semantically-enriched patterns.

- **Inconsistency:** There exist various forms of the SW data quality issues such as missing type properties (Paulheim & Bizer, 2013; Sleeman & Finin, 2013), incorrect or incomplete statements (Lehmann, Gerber, Morsey & Ngomo, 2012; Töpper, Knuth & Sack, 2012), invalid links to external resources (Halpin, Hayes, McCusker, McGuinness & Thompson, 2010; Paulheim, 2014), missing predicates (Lao & Cohen, 2010; Lao, Mitchell & Cohen, 2011), etc. However, the SW data quality issues are not limited to the mentioned categories and they are very diverse. Due to a large amount of RDF triples, there usually exists the possibility of assigning instances to incorrect classes in the ontology. For example, the DBpedia ontology defines a Royal class with two subclasses of BritishRoyalty and PolishKing for the royalties. *John I Albert*, king of Poland, has been assigned to BritishRoyalty class instead of PolishKing class in the DBpedia ontology. According to this motivation, Chapter 4 of this thesis works on an approach to tackle a SW data quality issue that is called Incorrect Class Assignment (ICA).

- **Incompleteness:** The SW adds semantics to the data by using ontological terminologies (De Nicola, Missikoff & Navigli, 2009). Ontologies are usually developed before data is published by data providers. So, they usually suffer from lack of completeness since they have not covered all aspects of specific domains. Class learning is one of the interesting topics in the area of ontology enrichment. In the SW, the most existing approaches rely on Inductive Logic Programming (ILP) techniques to learn new classes. Although the ILP is a helpful technique for developing logical disciplines, it usually requires counterexamples. To overcome the limitation of counterexamples, Chapter 5 of this thesis introduces a non-logical approach to extract new classes which are not defined in the ontology.

1.4 Research questions and objectives

The following describes the main research questions of the work to effectively address the motivations explained in Section 1.3. Research objectives have been also described corresponding to each question.

Research Question 1: How to automatically mine semantic association rules from RDF triples by utilizing instance-level and schema-level knowledge?

- **Objectives of Research Question 1:**
  - To develop an automated approach for discovering common behavioural patterns from RDF triples.
  - To highlight the importance of using relations and schema-level knowledge for generating semantically-enriched rules.

Research Question 2: How to analyse the correctness and incorrectness of relationships between instances and classes in the ontology?

- **Objectives of Research Question 2:**
  - To formally define a SW data quality problem called ICA.

  - To design and develop an approach that evaluates the status of instances that are correctly or incorrectly assigned to the classes in the ontology.

Research Question 3: How to automatically enrich schema-level knowledge by adding new classes which are not defined in the ontology?

- **Objectives of Research Question 3:**
  - To develop a non-logical approach that discovers new classes through mining instance-level and schema-level knowledge.
  - To place the generated classes in the hierarchical structure of an ontology.

1.5 Contributions

This thesis provides several contributions based on the research questions and objectives. The main contributions are described as follows:

- **SWARM**: Regarding to Research Question 1, an approach called Semantic Web Association Rule Mining (SWARM) is proposed to reveal common behavioural patterns among different types of entities. The behavioural patterns are associated with knowledge at the instance-level and schema-level. More precisely, the SWARM automatically mines semantic association rules by using knowledge encoded at the instance-level and schema-level. This approach takes advantage of relations and knowledge at the schema-level to generate semantically-enriched rules. The results highlight the necessity of using schema-level knowledge for interpreting rules. The SWARM also measures the quality of discovered rules by using class information of entities at the schema-level.
- **CAD**: Regarding to Research Question 2, an approach called Class Assignment

Detector (CAD) is proposed to tackle a SW data quality issue called ICA. This issue indicates incorrect assignments between instances and classes in the ontology. The CAD evaluates the correctness and incorrectness of relationships between instances and classes.

- **CEn**: Regarding to Research Question 3, an approach called Class Enricher (CEn) approach is proposed to mine knowledge at the instance-level and schema-level to discover new classes which are not defined in the ontology. The CEn enriches schema-level knowledge by discovering new classes. Additionally, the CEn approach identifies suitable places for generated classes in the hierarchical structure of the ontology.

1.6 Research methodology

The research methodology used in this thesis contains three main phases: Problem identification, Solution design, and Evaluation. The thesis follows the design science research methodology (Bichler, 2006; Rosemann & Vessey, 2008). Figure 1.2 depicts the research methodology process used in the thesis. As represented in the figure, the Problem identification phase is divided into two main steps: Identify the problem statement and Research questions. This phase offers a solid and important foundation for the further research process. In the Solution design phase, the design of study has been proposed to answer the research questions. In the Evaluation phase, the experiments are conducted, and the related techniques have been adopted into the methods. The appropriate Evaluation criteria are selected to assess the results obtained from the experiments. It is important to remind that the Literature review has been continuously conducted in each phase.

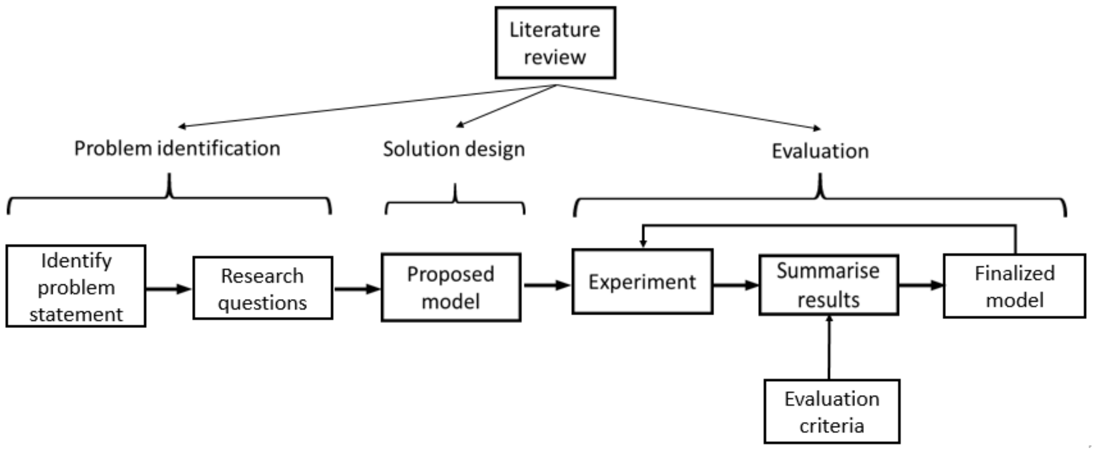

Figure 1.2: Research methodology

This thesis concentrates on three main research questions which are specifically covered in Chapters 3, 4, and 5. Each chapter follows the research methodology process shown in Figure 1.2. Several techniques are used at different stages throughout the thesis. To answer Research Question 1 in Chapter 3, the SWARM approach is proposed that adapts traditional association rule mining to RDF data. New quality factors are developed to measure the importance of extracted rules by considering the original concepts of rule factors in the traditional association rule mining. To answer Research Question 2 covered in Chapter 4, the CAD approach is designed to evaluate the correctness and incorrectness of relationships between instances and classes at the schema-level. To this end, the key features of classes are extracted by utilising information theory into RDF data. In this thesis, the CEn approach has also been developed in Chapter 5 inspired by the results obtained by the CAD. The CEn approach aims to satisfy the objectives of Research Question 3. It contains new algorithms to mine instance-level data by utilizing schema-level knowledge to create new classes that are not defined in the ontology. Several experiments have been conducted over real-world datasets to show the effectiveness of the SWARM, CAD, and CEn approaches.

1.7 Thesis structure

The remainder of the thesis is structured as follows:

Chapter 2 provides a literature review and background for this thesis. The chapter categorizes recent SW data mining studies into three main groups. The result of reviewing these studies reveal the main motivations for this thesis.

Chapter 3 concentrates on the objectives of Research Question 1. This chapter introduces an automated approach called SWARM that mines semantic association rules from the SW data. The chapter shows the necessity of using relations and schema-level knowledge in the mining process. The proposed approach has been tested over RDF-style datasets to demonstrate its effectiveness. The obtained results have been compared with one of the latest approaches to reveal the semantics of discovered rules by the SWARM.

Chapter 4 focuses on a SW data quality issue called ICA problem. The chapter presents the CAD approach and shows how the approach deals with the ICA problem. The chapter clearly explains how the CAD approach evaluates the correctness and incorrectness of relationships between instances and classes in the ontology. The experiments conducted over an RDF-style dataset shows the accuracy and effectiveness of the approach.

Chapter 5 deals with the incompleteness issue of ontologies in the SW. Specifically, this chapter introduces an approach called CEn to enrich ontologies by mining new classes. The chapter shows how to automatically mine instance-level and schema-level knowledge to generate new classes for an ontology. The effectiveness of this approach has been demonstrated by running experiments over an RDF-style dataset.

Chapter 6 discusses how the research questions have been answered by considering the objectives, summarises the contributions of the thesis and provides some interesting research directions for future work.

Chapter 2

Literature Review

The SW is an effort to make knowledge on the Web accessible and processable through its flexible architecture. Although the infrastructure of the SW gets rid of the rigidity of relational databases, it creates new challenges for mining and discovering knowledge needed for the users.

The aim of this chapter is to provide the related knowledge required to follow the research questions and objectives described in Chapter 1. To this end, the chapter presents a detailed review of research in mining and learning from the SW. Sections 2.1-2.3 classify the existing studies into three main categories including Frequent pattern mining, SW data quality improvement, and ontology enrichment. Finally, Section 2.4 summarises the chapter and shows how motivations of this thesis are identified by the limitations in the recent studies.

2.1 Frequent pattern mining

Over recent decades, there has been an increasing demand in developing data mining methods for extracting frequent patterns from the SW data (Chomboon, Kaoungku, Kerdprasop & Kerdprasop, 2014). In the context of the SW, most existing frequent pattern mining methods can be classified into three main categories, i.e., Frequent subgraph mining, Logical rule mining, and Association rule mining.

2.1.1 Frequent subgraph mining

The concept of frequent subgraph mining focuses on discovering substructures that frequently occur in the graph-structured datasets (Kuramochi & Karypis, 2001; Huan, Wang & Prins, 2003; Chi, Muntz, Nijssen & Kok, 2005). This is an important topic since data can be represented as a graph in various fields such as chemical molecules and social networks.

Several studies have been conducted over the SW data since it can be expressed as a directed labeled graph. Work by Ramakrishnan et al. (2005) proposed a subgraph mining method to develop weighting mechanisms extracted from semantic links in the SW graphs. This method evaluates the quality of extracted subgraphs by using path ranking metrics. To improve the process of extracting subgraphs, Kasneci et al. (2009) introduced MING algorithm that defines informativeness criteria to build a random surfer model for ranking RDF query results based on their informativeness.

To speed up the query search results and reduce the volume of storage spaces, the summarisation techniques are developed to generate smaller graphs. On this subject, a study conducted by Basse et al. (2010) addressed the issue of extracting a compact representation of RDF graphs. The method extended the gSpan algorithm proposed by Yan et al. (2004) into a form of directed labeled graphs along with DFS (Depth-First Search) code as a canonical representation of RDF graphs. Zhang et al. (2012) proposed another method for mining link patterns in the LOD to enhance the scalability level. In this study, before the mining process, a Typed Object Graph collects instances for the sake of scalability. To compare with the above methods, DIVERSUM is another summarisation method that achieves higher recall scores (Sydow, Pikuła &

Schenkel, 2013). DIVERSUM concentrates on the diversification in the graphical entity summarisation.

As explained in Chapter 1, SPARQL is a SW query language that aims to discover proper subgraphs from the LOD cloud based on the users' inquiries. Researchers have been recently developing methods to optimize the results obtained by SPARQL engines. HiBISCuS is a method that transforms SPARQL queries into directed labeled hypergraphs (Saleem & Ngomo, 2014). HiBISCuS filters out irrelevant results by considering the joins' types in the queries. It is important to mention that the query runtime of HiBISCuS outperforms DARQ (Quilitz & Leser, 2008), SPLENDID (Görlitz & Staab, 2011), and FedX (Schwarte, Haase, Hose, Schenkel & Schmidt, 2011) methods. Similarly to HiBISCuS, a query rewriting-based method utilizes the interconnection topology between RDF systems to filter out irrelevant results (Peng, Zou, Özsu & Zhao, 2018). To improve the semantics of retrieved results, KAT used the context of each input keyword (Wen, Jin & Yuan, 2018). The KAT is developed based on two indexing mechanisms called keyword index and graph index. The keyword index uses RDF class information to reduce keyword ambiguity, while the graph index is created to speed up the graph mining process.

Although the above algorithms are interesting in the area of frequent subgraph mining, they usually hide content associations in the SW graph. Because they are concerned with frequent substructures. More precisely, these algorithms do not often consider knowledge encoded at the schema-level for extracting semantically frequent content in the SW graph. In contrast with the above algorithms, this thesis aims to not only take instance-level data into account but also schema-level knowledge to discover semantically-enriched patterns from the SW data.

2.1.2 Logical rule mining

The ILP is an area of machine learning that employs logic programming as a formal representation. The ILP usually requires counterexamples (Muggleton & De Raedt, 1994). The very first step in ILP-based methods is to provide a set of positive and negative facts along with some background knowledge. On this line of research, ALEPH is one of the well-known ILP-based systems that is implemented in the Prolog (Muggleton, 1995). WARMR is another ILP-based system that mines patterns from relational databases by using conjunctive queries (Dehaspe & Toivonen, 1999, 2001). WARMER is an extended version of WARMR that uses a declarative language, and it supports a broader range of conjunctive queries to mine frequent patterns from relational databases (Goethals & Van den Bussche, 2002).

ILP is also a well-known technique for mining frequent patterns from the SW data (d'Amato, Tettamanzi & Minh, 2016). JÓzefowska et al. (2010) developed an algorithm to discover frequent patterns from the SW. Galárraga et al. (2013) proposed a multi-threaded method called AMIE for mining Horn rules from RDF-style KBs. This method concentrates on mining Horn rules among predicates such as $motherOf(m,c) \wedge marriedTo(m,f) \Rightarrow fatherOf(f,c)$. AMIE's quality factors (i.e., support and confidence) only consider knowledge at the instance-level and they remove the *rdf:type* properties from the ontology. In order to reduce the number of generated Horn rules, the AMIE mines closed rules, i.e., each of the variables in the predicates should have appeared at least twice. For example, in the mentioned rule, the variable m has been shared between *motherOf* and *marriedTo* twice. AMIE also introduced a new measure called Partial Completeness Assumption (PCA). PCA is defined in this way that if a dataset contains some triples on a subject s with a predicate p, then it knows all the possible triples of s with the predicate p. Furthermore, a quality factor called the PCA confidence was introduced in AMIE. As explained, PCA is restricted to a clear and complete ontological knowledge base which does not exist in many SW systems. Additionally, AMIE is a multi-threaded method with high memory overheads. Galárraga et al. (2015) extended AMIE to AMIE$^+$ by using pruning and query rewriting techniques. Mining Horn rules from the SW data has also been studied in (B. Yang, Yih, He, Gao & Deng, 2014). EDMAR is another method that takes advantage from problem-aware genetic operators to discover associations in the form of Semantic Web Rule Language (SWRL) rules (Tran, d'Amato, Nguyen & Tettamanzi, 2017; Tran, d'Amato, Nguyen & Tettamanzi, 2018).

Although ILP is a useful method for developing logical disciplines by relying on the reasoning and predefined statements, it is not sufficient for representing concepts that need non-logical backbones. ILP usually requires counterexamples which is a limitation of this method. In the SW, most existing ILP-based methods consider instance-level data for evaluating the interestingness of frequent patterns by using quality factors (Lisi & Esposito, 2005; L. A. Galárraga, Teflioudi et al., 2013; B. Yang et al., 2014; L. Galárraga et al., 2015). These methods disregard knowledge encoded at the schema-level, i.e., by removing *rdf:type* properties from the RDF-style datasets. Consider *isCitizenOf*(*x,y*) ⇒ *livesIn*(*x,y*) as a frequent pattern discovered by AMIE (L. A. Galárraga, Teflioudi et al., 2013). The AMIE ignores *rdf:type* properties from the ontology for measuring the quality of the result. While, instances that support the above pattern might belong to different classes in the ontology. In the context of the SW data, it is not rational to interpret discovered patterns by ignoring the knowledge embedded at the schema-level. Using class information is an essential fact for adding another level of semantics to the results. The above limitations motivate us to show how using schema-level knowledge besides instance-level data can generate semantically-enriched patterns from the SW data.

2.1.3 Association rule mining

The concept of association rule mining was first introduced by Agrawal et al. (1993). Let $I = \{i_1, i_2, ..., i_n\}$ be a set of items and $D = \{t_1, t_2, ..., t_m\}$ be a set of transactions. Each transaction contains a subset of items in *I*. An association rule represents a frequent pattern of the occurrence of some items in the transactions. In addition, association rules generally reveal behavioural patterns of some particular entities in the datasets (Han, Pei & Yin, 2000; Han, Cheng, Xin & Yan, 2007). For example, in the traditional shopping basket problem, the rule {*butter, bread*} ⇒ {*milk*} shows a common behaviour of customers. Namely, if a customer buys *butter* and *bread* together, she is likely to buy *milk* as well.

Association rule mining is not only limited to tabular data. The concept of generalized association rule was proposed to discuss how to mine frequent patterns from taxonomies (Srikant & Agrawal, 1995). The purpose of this method is to extend itemsets by considering all the ancestors of each item in a taxonomy and consequently mining generalized association rules. The extracted rules show details at different granularities. This method has the main drawback that is about transactions length overhead that negatively impacts the results. Different optimization methods have been proposed to improve this issue (Tsur et al., 1998; Wei & Chen, 1999; Chen, Wei & Kerre, 2000; L. Yang, 2005). On this subject, Diff_ET and Diff_ET2 are two novel algorithms that efficiently extract generalized association rules in the taxonomy (Tseng, Lin & Jeng, 2008).

Discovering frequent patterns has also been studied for assessing users' behaviours in the Web log files. Different methods are proposed to predict Web pages that a user is likely to visit through analysing Web log files reviewed by the user (Géry & Haddad, 2003). In this respect, the FS-Miner is an efficient system designed for mining frequent sequences from Web log files (El-Sayed, Ruiz & Rundensteiner, 2004). The performance of FS-Miner shows that the system scales linearly that is suitable for increasing the size of input data. Similar strategies also discussed in (Takama & Hattori, 2007; Singh, Kumar & Maurya, 2014; Raphaeli, Goldstein & Fink, 2017; Shanthi, 2017; Kaur & Aggarwal, 2018).

Traditional association rule mining has also been adapted to the SW data for discovering frequent hidden patterns. Nebot and Berlanga (2012) proposed a rule mining method for RDF-based medical data. In the proposed method, transactions are generated by using mining patterns developed by SPARQL queries. This method semi-automatically obtains the desired terminology for generating items and transactions for the association rule mining process. This method relies heavily on the domain experts. Namely, users should have background knowledge of the vocabularies used in the ontology. Mining Configurations proposed by Abedjan and Naumann (2013a, 2014) is an automatic method that performs based on the depth-first search using the FP-Growth algorithm (Han et al., 2000) to automatically discover association rules from RDF data. Mining Configurations improves the limitations in the (Nebot & Berlanga, 2012). This method discovers dependencies between entities by using six different configurations. The method claims that is useful in various use cases such as schema discovery, basket analysis, clustering, range discovery, topical clustering, and schema matching. In this method, any part of Subject-Predicate-Object (SPO) statement can be considered as a *context*, which is used for grouping one of the two remaining parts of the statement as the *target* of mining. Mining Configurations generates three forms of rules including $s_i \Rightarrow s_j$ (mining subjects), $p_i \Rightarrow p_j$ (mining predicates), and $o_i \Rightarrow o_j$ (mining objects). Note that the method generates association rules by mining instance-level data and it does not consider the schema-level knowledge in its process. Additionally, the rule quality factors (i.e., support and confidence) only consider entities at the instance-level. SAG is another method that used SPARQL commands to mine and classify association rules from the RDF data without considering schema-level knowledge (Tsay, Sukumar

& Roberts, 2015).

Ignoring ontological knowledge can have negative impacts on interpreting the rules. For example, Mining Configurations claims that this method can be used in the schema discovery scenarios (Abedjan & Naumann, 2013a, 2014). In the Mining Configurations method, an association rule like *associatedBand*, *instrument* ⇒ *associtedMusicalArtist* shows a schema for musicians. Although this method detects some interesting associations, in some cases, it is not able to express a certain schema for the rules. For example, the schema of rule *influencedBy* ⇒ *influenced*, *birthPlace*, *deathPlace* is not recognizable by Mining Configurations method. Because the predicates in the above rule are common among different classes in the ontology. For example, *influencedBy* is a common predicate among Philosopher, Saint, and Writer classes in the DBpedia ontology.

To overcome these limitations, this thesis aims to reveal the importance of utilizing schema-level knowledge in the process of mining frequent patterns.

2.2 SW data quality improvement

Because of the distributed nature of the SW, RDF-style KBs usually contain noisy and inconsistent data that create different problems known as SW data quality issues. Different methods are proposed to tackle the SW data quality issues driven by the diversity and complexity of KBs. The main purpose of these methods is to refine and re-engineer the KBs. The following describes some SW data quality issues that frequently occur in the popular RDF-style KBs such as (i) missing type properties, (ii) incorrect or incomplete statements, (iii) incorrect links to external resources, and (iv) missing predicates.

2.2.1 Missing type properties

In the SW, a relationship between an instance and corresponding classes in the ontology is provided by an *rdf:type* property. In other words, *rdf:type* property indicates a class that an instance belongs to. In this respect, KBs usually suffer from missing *rdf:type* properties since the process of distributing the SW data is controlled by users.

On this line of research, Tìpalo is an automatic algorithm for predicting missing type properties in the DBpedia datasets by interpreting natural language description extracted from Wikipedia abstracts (Gangemi et al., 2012). To exploit more potentials provided by the SW technologies, a method proposed by Giovanni et al. (2012) takes advantage from links within Wikipedia infoboxes to infer missing type properties. Work by Paulheim and Bizer (2013) developed a heuristic type inference mechanism called SDType to handle missing type properties. It is worth mentioning that the results obtained by the SDType show that at most 63.7% of the SW data have complete type declarations in the DBpedia and at most 53.3% in the YAGO. To enhance the performance of SDType, a method called SLCN predicts missing *rdf:type* property in the RDF-style KBs (Melo, Völker & Paulheim, 2017). Note that SLCN can perform and scale better to compare with SDType method (Paulheim & Bizer, 2013). Sleeman and Finin (2013) proposed a supervised machine learning algorithm to predict the missing type of instances by mapping their attributes into a common attribute space. Kellou-Menouer and Kedad (2015) also presented a clustering-based method to infer missing *rdf:type* properties under incomplete RDF-style KBs. It is important to know that the reviewed studies focus on predicting missing types for object property values in the SW data.

There also exists the possibility of missing types for datatype properties. Gunaratna et al. (2016) proposed a method that identifies missing types for datatype property values by semantic matching of terms to the class labels. For example, consider an RDF triple (*Barack Obama*, *shortDescription*, *"44 president of the United States"*

^^xsd:string) from DBpedia. The method assigns the literal (*"44 president of the United States"* ^^xsd:string) to the President class by mining the semantic of literal.

2.2.2 Incorrect or incomplete statements

There is another SW data quality issue that refers to incorrect or incomplete object values of RDF triples. Consider triple (*Rodrigo Salinas*, *birthPlace*, *Puebla F.C.*) shared by DBpedia. The DBpedia provides an invalid object value for *Rodrigo Salinas* who is a Mexican football player. Instead of sharing a city or a country name, the *Puebla F.C.* stadium has been extracted from Wikipedia abstracts.

According to this issue, Defacto is a system built based on a lexical library of properties that frequently checks DBpedia datasets to analyse the validity of RDF triples (Lehmann et al., 2012). Research completed by Töpper et al. (2012) focused on modifying existing errors occurred in the object values of RDF triples by exploiting domain and range properties as well as class disjointness axioms. On this line of research, an outlier detector method is proposed to identify wrong numerical object values in the RDF-style KBs (Fleischhacker, Paulheim, Bryl, Völker & Bizer, 2014). Debattista et al. (2015; 2016) proposed a distance-based clustering method to assess the possibility of identifying incorrect RDF statements in the SW. To make existing methods robust, Dongo et al. (2017) proposed a method that combines lexical analysis and semantic analysis together to derive datatype property in four main steps: (I) analysing existing range property, (II) analysing object lexical-space, (III) semantic analysis of the predicate name, and (IV) generalizing of numeric datatypes.

In the SW, predicates are not only used for linking entities to other entities but also to literals. Many datatype properties of RDF triples have not been entered in the KBs due to a large amount of data. Rizzo et al. (2016) proposed a method that combines predictive clustering trees and terminological regression trees to predict unknown values.

The proposed method also adopts SWRL to represent extracted rules concerning with numeric attributes.

2.2.3 Invalid links to external resources

By increasing the number of the SW data over the LOD cloud, faulty links to the external KBs are inevitable. This particular problem occurs when the association between the subjects and objects is inaccurate (Acosta et al., 2016). In the OWL terminology, an instance linked by *owl:sameAs* property refers to the same real-world instance in another knowledge base (Ding, Shinavier, Shangguan & McGuinness, 2010). Such instances share the same facts across the LOD cloud.

A survey conducted by Halpin et al. (2010) assessed how accurate *owl:sameAs* property is being used (and misused) over the LOD cloud. One of the most prominent systems is called Silk that links entities through developing a heuristic method (Volz, Bizer, Gaedke & Kobilarov, 2009). This method has been extended by using blocking strategies (Isele, Jentzsch & Bizer, 2011) and genetic algorithms (Isele & Bizer, 2012). To compare with Silk, LINDA is a distributed link discovery method based on Hadoop algorithms which can be scaled for the very large datasets (Böhm, de Melo, Naumann & Weikum, 2012). LiQuate is a tool that uses Bayesian Networks and rule-based systems to evaluate the quality of links in the LOD cloud (Ruckhaus, Baldizán & Vidal, 2013). Similarly to LiQuate, Paulheim (2014) developed a method to discover faulty links between datasets by using multi-dimensional outlier detection techniques. In this regard, a metric-driven method improves the quality of link analysis by using three measures including internal linking, external linking, and link-ability from other resources (Yaghouti, Kahani & Behkamal, 2015). A method proposed by Spahiu et al. (2016) evaluates the quality of *owl:sameAs* property and discovers similar links by using similarity metrics.

2.2.4 Missing predicates

Publishing the SW data is a very error-prone process due to a large amount of data. Predicting missing properties (i.e., predicates) is another valuable research topic in the SW field. Consider *Barack Obama* as a subject and *Honolulu* as an object of an RDF triple. Here the question is that how to predict *birthPlace* property by mining existing RDF data in the SW.

According to this quality issue, a path ranking algorithm extended the idea of using random walks for predicting links in multi-relational knowledge graphs (Lao & Cohen, 2010; Lao et al., 2011). Krompaß et al. (2015) proposed a type-constraints based method to extract the semantic relationships among instances in a closed-world assumption. It is worth mentioning that the method performs better than other methods (Nickel, Tresp & Kriegel, 2011; Bordes, Usunier, Garcia-Duran, Weston & Yakhnenko, 2013; Dong et al., 2014). Last but not least, Cao et al. (2016) proposed a supervised method to learn weights of meta paths to build a link prediction model for RDF-style KBs.

As reviewed above, there exist different kinds of the SW data quality issues such as missing type property, incorrect or incomplete statements, incorrect links to external resources, and missing predicate. However, the SW data quality issues are more challenging than our expectations. This thesis works on a SW data quality issue called ICA. The motivation behind the ICA issue is inspired by observing some inconsistencies between instances and classes in the DBpedia dataset. The ICA issue indicates incorrect assignments between instances at the instance-level and classes in the ontology.

2.3 Ontology enrichment

By developing the SW technologies for supporting the Web 2.0 infrastructure, the number of KBs are also growing fast. An RDF-style knowledge base not only contains instance-level/RDF triples but also schema-level/ontology. An ontology is normally static since the vocabularies describing domains do not often change, while the instance-level is very dynamic because it is continuously updated (Tulasi & Rao, 2014; Shah & Jain, 2014; Dou, Wang & Liu, 2015). On the other side, ontologies can be incomplete since they are usually created before using in the real-world scenarios. In this respect, ontology enrichment is an interesting line of research that is mainly about automated or semi-automated schema, class and property learning. The rest of this section reviews most common opening studies in the area of ontology enrichment including (I) Schema learning, (II) Property learning, and (III) Class learning.

2.3.1 Schema learning

An ontology presents the conceptualization of specific domains while the assertional knowledge is usually provided by RDF triples for classes in the ontology. With the explosive growing the SW data over the LOD cloud, the problem is that traversing large ontologies consumes more time and space. In this respect, schema learning is a useful mechanism to overcome this limitation. The main objective of schema learning studies is to construct a more unified ontological structure by mapping different schemas to each other.

To this intent, a hierarchical-based clustering method is developed based on a set of measures to assess the similarity between metadata in the ontologies (Maedche & Zacharias, 2002). Similarly to Maedche and Zacharias (2002), a method proposed by de Mantaras and Saitia (2004) used clustering techniques by defining formal concepts and context vectors to build taxonomies. To obtain a better level of feasibility in the process of schema mapping, a K-means clustering based method maps different ontological structures to each other (Esposito, Fanizzi & d'Amato, 2007).

It is interesting to know that some methods have been developed to directly map the schemas of databases to ontological structures. On this line of research, An and Topaloglou (2008) proposed a specification for the validity of a semantic mapping between the schemas of databases and ontologies. To improve the efficiency level, Khattak et al. (2012) proposed another method that takes advantage from change history to diminish the time needed for mapping among ontologies. Similar statistical methods are also studied in the (Völker & Niepert, 2011; Dos Reis, Pruski & Reynaud-Delaître, 2015).

Since traversing ontological structures is a very time-consuming task, different methods are focused on partitioning ontologies for the sake of scalability. Ahmed et al. (2015) implemented a clustering-based method to partition OWL ontologies by using semantic similarity measures. Similar partitioning methods are also studied to deal with the memory consumption (Suh & Gaddam, 2011; Saruladha, Aghila & Sathiya, 2012).

One of the main tasks in the schema mapping process is to analyse the similarity between classes. In this respect, CWCONS is a method that discovers equivalence relation between classes (Yin, Gu & Hou, 2016). The motivation of CWCONS is to categorise the nodes of different ontology trees into two groups including classification nodes and the concept nodes. To make the results achieved by CWCONS robust, Gao et al. (2017) studied how to obtain an optimal distance measure in the ontology mapping. It is interesting to know that MAPSOM demonstrated how ontological descriptions facilitate interoperability between a data model and new data sources in the process of ontology mapping (Jirkovskỳ, Kadera & Rychtyckyj, 2017).

Although the above studies make some valuable progress in the field of ontology mapping, it still suffers from several challenging issues, including heavy computational overheads of ontology mapping and lacking automated methods to fully perform the mapping process.

2.3.2 Property learning

A property/predicate plays an essential role in representing an RDF triple. Because a predicate expresses a meaningful relationship between entities. The following reviews recent studies for mapping synonym properties between RDF-style KBs for ontology enrichment.

In the SW, although property learning is one of the most crucial steps for enriching KBs, little effort has been put forward. Many RDF-style KBs overlap with each other not only in their instances but also classes and properties defined in the ontologies. PARIS is an automatic alignment method that aims to align and interlink SW-based ontologies (Suchanek, Abiteboul & Senellart, 2011). PARIS aligns not only instances and classes but also properties. The experiments show that PARIS is able to discover synonym properties between YAGO and DBpedia such as *y:isCitizanOf* ⊆ *dbp:nationality*, *y:isMarriedTo* ⊆ *dbp:spouse*, and *dbp:award* ⊆ *y:hasWonPrize*. There are also many synonym facts in different RDF-style KBs such as (*Washington*, *CapitalCityOf*, *U.S.*) and (*D.C.*, *IsCapitalOf*, *United States*). Aligning such facts between two KBs have been studied in (Parundekar, Knoblock & Ambite, 2010; L. A. Galárraga, Preda & Suchanek, 2013; Amarilli, Galárraga, Preda & Suchanek, 2014; L. Galárraga, Heitz, Murphy & Suchanek, 2014). Mining synonym properties not only discovers groups of equivalent relations but also helps recommender systems to suggest a wide range of predicates to the SW data providers. A recommender system proposed by Abedjan and Naumann (2013b) used association rules to extract synonym properties from the DBpedia ontology. Then, the matched synonyms are exploited for the property development.

Although the results of such methods are applicable in the SW applications, the foundation of these methods are not sophisticated enough to align multiple KBs (i.e., more than two KBs) to discover synonym properties.

2.3.3 Class learning

A class/concept plays a fundamental role in the ontology enrichment. A class is an abstract notion that contains a set of things with similar attributes. Since ontologies usually suffer from lack of completeness, class learning helps to extract hidden concepts. The following reviews how class learning methods assist ontologies to be completed in the SW field.

In the SW technologies, Description Logic (DL) has been exploited as the foundation of OWL (McGuinness, Van Harmelen et al., 2004; d'Amato, Fanizzi & Esposito, 2010). It is necessary to mention that the concept of DL originates from ILP. In the OWL, axioms are defined to express different relationships between classes. There exist various semantics for representing relations among Classes (C) such as Subsumption (C_i, C_j), Equivalence (C_i, C_j), Disjoint (C_i, C_j), etc. The Subsumption (C_i, C_j) shows that C_i is a subclass of C_j; Equivalence (C_i, C_j) indicates that there exists a binary relation between C_i and C_j that can express reflexive, symmetric, and transitive properties; Disjoint (C_i, C_j) also shows that C_i and C_j are completely two different classes in the ontology.

On this subject, Iannone et al. (2007) investigated solutions for the class induction by a semi-automatic DL-based method. In comparison to Iannone et al. (2007), an automated method proposed by Bühmann et al. (2016) is a DL-Learner that encompassed a set of algorithms for class learning by using a refinement operator. The refinement operator contains two main operators including downward refinement operator to extract a set of classes and upward refinement operator which extracts a set of general classes for the input data. Fleischhacker et al., (2011) proposed a method to inductively learn disjointness axioms. They discussed multiple strategies such as learning the correlation between two classes by using the number of common instantiations between pairs. Since RDF-style KBs usually contain inconsistent data, Töpper et al. (2012) shows the benefit of disjoint axioms for discovering inconsistencies. The authors proposed a method to detect those class pairs that have similarity scores below a fixed threshold. Such pairs are considered as disjoint classes. BelNet is a logical-based method that specifically learns subsumption relationships from the schema-level by using DL and Bayesian Network under the incomplete KBs (Zhu et al., 2013). There exist similar ILP-based methods that are successfully applied in the practical applications to extract OWL axiom expressions (Lehmann, 2009; Hellmann, Lehmann & Auer, 2009; Suchanek et al., 2011; d'Amato, Bryl & Serafini, 2012; L. A. Galárraga, Preda & Suchanek, 2013; D. Zhang, Yang, Wang, Wang & Zhao, 2016).

Since different schemas are connected by the SW technologies, a method proposed by Bühmann and Lehmann (2013) discovered frequent axiom patterns in different ontologies and transformed them into SPARQL query patterns. Then, the query patterns are applied to other datasets to enrich them with new axioms. To enhance the process of mining axioms from different ontologies, Li et al. (2015) suggested another method to discover axioms by splitting the dataset into several blocks based on disjoint properties.

To the best of knowledge, the existing class learning methods are mainly based on ILP techniques that usually work with a set of counterexamples. To overcome the limitation of counterexamples, this thesis introduces a non-logical method that generates new classes which are not defined in the ontology through mining instance-level and schema-level knowledge.

2.4 Summary

This chapter provides a detailed view of recent learning and mining methods in the SW field. The studies are sorted into three main categories including (I) Frequent pattern mining, (II) SW data quality improvement, and (III) Ontology enrichment. By reviewing related work, several shortcomings and directions are identified as follows.

- In the process of discovering frequent patterns, the existing SW data mining methods usually do not take schema-level knowledge into account. In the proposed methods, the extracted frequent patterns usually use knowledge at the instance-level without considering this fact that instances might belong to different classes at the schema-level. This thesis shows that the extracted patterns do not reveal explicit semantics without utilizing schema-level knowledge.
- By explosive growing the SW data, various quality issues are also emerging since the process of publishing data is managed by the SW communities. By detecting inconsistencies between instances and classes in the DBpedia ontology, this thesis deals with a SW data quality issue called ICA.
- While the number of RDF-style KBs is increasing, the enrichment of ontology schemata remains as a challenge. In this line of study, class learning is a topic that concentrates on extracting concepts from the SW data. In the SW, the existing methods are usually developed by using ILP techniques are suitable for logical scenarios with a set of counterexamples. To extract new classes and get rid of counterexamples, this thesis concentrates on a non-logical approach for class learning.

In the area of mining and learning from the SW data, several research gaps have been identified from the literature review which are pointed above. The following chapters specifically discuss challenges and propose approaches to deal with them.

Chapter 3

Mining Semantic Association Rules From RDF Data

Although the SW adds semantics to RDF triples by providing ontological vocabularies, this potential is not exploifted very well for expressing associations among triples. Traditional association rule mining is a data mining method that has been adapted into the SW for extracting frequent patterns. Most existing SW-based methods designed for discovering association rules usually define quality factors (e.g., support and confidence) by only considering instance-level data. In fact, these methods disregard knowledge embedded at the schema-level. In this regard, the semantics behind discovered rules is not explicitly interpretable when entities have different types at the schema-level.

Mining Configurations is a SW-based mining method for discovering association rules (Abedjan & Naumann, 2013a). The method measures the quality of rules without using ontological knowledge. More precisely, Mining Configurations disregards *rdf:type* properties, and it just focuses on mining instance-level data. The studies conducted in this chapter shows that Mining Configurations method is not able to provide explicit semantics for association rules in the SW.

This chapter introduces an effective rule mining approach called Semantic Web Association Rule Mining (SWARM) that automatically mines and generates semantically-enriched rules from RDF data. Different with most existing studies, the SWARM approach exploits *rdf:type* and *rdfs:subClassOf* properties defined in the ontology to enrich association rules. The discovered rules reveal common behavioural patterns associated with knowledge at both instance-level and schema-level. This chapter illustrates the benefits of using relations and schema-level knowledge by comparing the results with the Mining Configurations.

The rest of this chapter is structured as follows. Section 3.1 provides a motivating example to show the importance of considering relations and schema-level knowledge in the process of mining association rules. The SWARM's framework, definitions, and algorithms are presented in Section 3.2. Section 3.3 explains the experimental setup and results. The extensibility of the SWARM is also discussed in Section 3.4. Finally, Section 3.5 summarises the whole chapter and highlights the contributions of the SWARM.

3.1 Motivation

The most important thing here is to provide a clear description of some instance-level and schema-level knowledge in the discovered association rules. By considering RDF triples in Table 3.1, an ideal way to represent semantics in an association rule is shown as follows:

$$\{\textit{Person}\}\text{: } (instrument, \textit{Guitar}) \Rightarrow (occupation, \textit{Songwriter})$$

The above rule shows that most of the time people who play a musical instrument (e.g., guitar) are probably songwriters. Mining such regularities help us to achieve a better understanding of the SW data. For example, we can find out that people with similar skills usually tend to join to a particular community; people often communicate with those speaking the same language; people born in the same region usually tend to collaborate with each other, and so on. The goal of the SWARM is to extract such behavioural patterns from RDF-style KBs.

Table 3.1: Some RDF triples from the DBpedia dataset

Triples	**Subject**	**Predicate**	**Object**
t_1	*John Lennon*	*instrument*	*Guitar*
t_2	*John Lennon*	*spouse*	*Yoko Ono*
t_3	*John Lennon*	*occupation*	*Songwriter*
t_4	*Yoko Ono*	*birthplace*	*Tokyo*
t_5	*George Harrison*	*instrument*	*Guitar*
t_6	*George Harrison*	*occupation*	*Songwriter*
t_7	*Jimmy Carter*	*office*	*President of the USA*
t_8	*Jimmy Carter*	*party*	*Democratic*
t_9	*Bill Clinton*	*office*	*President of the USA*
t_{10}	*Bill Clinton*	*party*	*Democratic*
t_{11}	*George W. Bush*	*office*	*President of the USA*
t_{12}	*George W. Bush*	*party*	*Republic*
t_{13}	*John Lennon*	*rdf:type*	*dbo:Person*
t_{14}	*George Harrison*	*rdf:type*	*dbo:MusicalArtist*
t_{15}	*George Harrison*	*rdf:type*	*dbo:Person*
t_{16}	*Jimmy Carter*	*rdf:type*	*dbo:Person*
t_{17}	*Bill Clinton*	*rdf:type*	*dbo:Person*
t_{18}	*George W. Bush*	*rdf:type*	*dbo:Person*
t_{19}	*Yoko Ono*	*rdf:type*	*dbo:Person*

In order to boost and enrich the semantics of rules, the SWARM considers *rdf:type* and *rdfs:subClassOf* properties at the schema-level. In the OWL vocabularies, *rdf:type* is basically a property that ties an instance to a class in the ontology; *rdfs:subClassOf* is also used to show that one class is a subclass of another class in the ontology. As seen in Table 3.1, both *John Lennon* and *George Harrison* are guitarists and songwriters. Consider Figure 3.1 as a small fragment of the DBpedia ontology (3.8). *George Harrison* is an instance of Musical Artist class while *John Lennon* belongs to the Person class. Regarding to the concept of hierarchy in the ontology, if the Musical Artist class is a subclass of the Artist class and the Artist class is a subclass of the Person class, then *George Harrison* belongs to the Person class as well. However, *John Lennon* is not indicated to be an instance of the Musical Artist class.

In the SW, it is not rational to interpret the discovered rules by ignoring knowledge embedded at either the instance-level or the schema-level. Unfortunately, most existing association rule mining methods for the SW data only focus on discovering frequent patterns at the instance-level. Hence, the major motivation of the SWARM is to overcome this drawback by using the instance-level and schema-level knowledge.

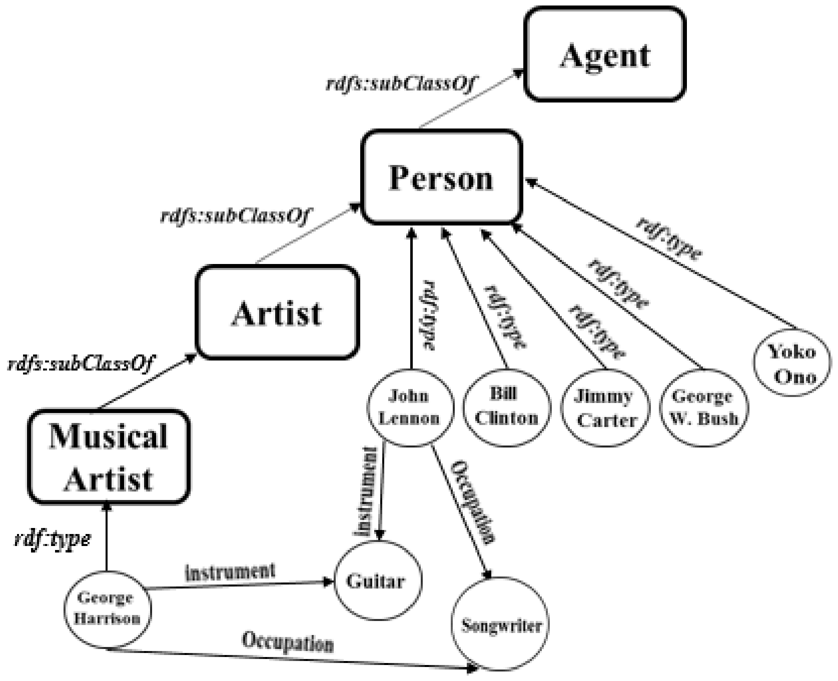

Figure 3.1: A fragment of the DBpedia ontology

3.2 The Semantic Web Association Rule Mining (SWARM) approach

The overall framework of the SWARM is shown in Figure 3.2. It contains two major modules: Pre-processing module and Mining module. RDF triples are automatically processed by the Pre-processing module which contains two sub-modules: Semantic

Item Generation and Common Behaviour Set Generation. The Mining module receives Common Behaviour Sets to generate Semantic Association Rules. The SWARM approach evaluates the quality of rules by using *rdf:type* and *rdfs:subClassOf* properties in the ontology. The proposed rule quality factors not only considers knowledge at the instance-level but also at the schema-level.

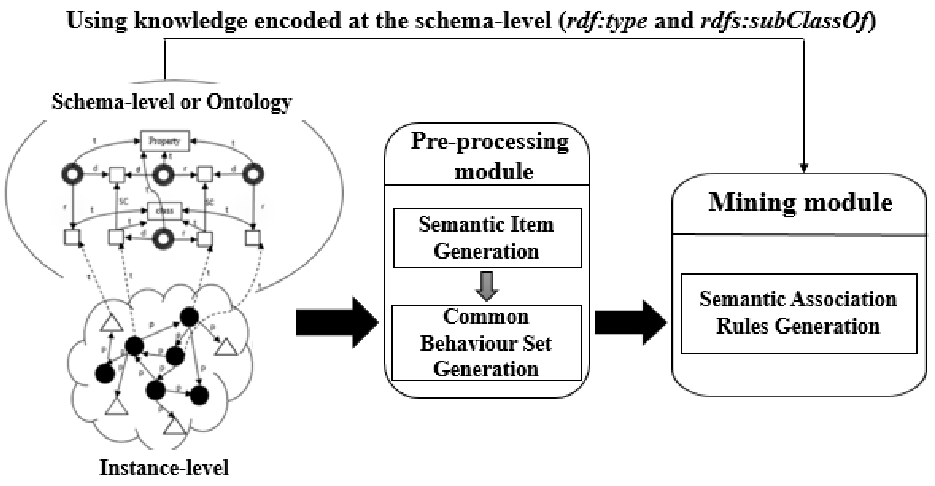
Using knowledge encoded at the schema-level (rdf:type and rdfs:subClassOf)
Schema-level or Ontology
Property
class
Instance-level
Pre-processing module
Semantic Item Generation
Common Behaviour Set Generation
Mining module
Semantic Association Rules Generation

Figure 3.2: The SWARM framework

3.2.1 Pre-processing module

Traditional association rule mining algorithms are mainly suited for homogeneous repositories, where items and transactions play significant roles in the mining process (Han et al., 2000). However, most SW data are not transactional data, and there exist no items or transactions. In this respect, we need to model such notions to discover associations from the SW data.

As mentioned earlier, the assertion of an RDF triple, i.e., (subject, predicate, object) indicates a meaningful relationship between a subject and an object provided by a predicate. Namely, a triple can also be considered as the description of one particular behaviour of entities. For example, if we consider the subject of $t1$, i.e., *John Lennon*, in Table 3.1 as an entity (i.e., instance), then the other two elements in the triple (i.e., *instrument, Guitar*) can be considered as a particular behaviour of *John Lennon*. Based on this idea, the concept of Entity Behaviour is defined as follows.

Definition 3.3.1. (*Entity Behaviour*). Given an RDF triple t_i, an Entity Behaviour derived from t_i is a 2-tuple, i.e., eb_i = (e_i, pa_i). e_i is an Entity which can be either the subject (s) or the object (o) of the triple. pa_i is the Pair of eb_i. It indicates a behaviour taken by e_i. Corresponding with the content in e_i, pa_i contains the combination of predicate-object or predicate-subject, i.e., (p, o) or (p, s).

Based on Definition 3.3.1., the following defines the concepts of Semantic Item and Common Behaviour Set to show how the SWARM extracts common behaviours among entities.

Semantic Item Generation

As seen in Table 3.1, the subjects in the $t1$ and $t5$, i.e., *John Lennon* and *George Harrison*, share a common activity, i.e., (*instrument, Guitar*). Namely, playing guitar is a common behaviour taken by this group of entities, i.e., *John Lennon* and *George Harrison*. In the SWARM approach, such combinations are known as Semantic Item.

Definition 3.3.2. (*Semantic Item*). A Semantic Item si_j is a 2-tuple, i.e., si_j = (es_j, pa_j). es_j is an Element Set of si_j. It contains a list of subjects or objects, i.e., $\{s_1, s_2, ..., s_n\}$ or $\{o_1, o_2, ..., o_n\}$. pa_j is a Pair of si_j. Corresponding with the content in es_j, pa_j contains a combination of predicate-object or predicate-subject, i.e., (p, o) or (p, s).

According to Definition 3.3.2., triples in a triple store can be converted to a set of Semantic Items i.e., SI=$\{si_1$, $si_2, ..., si_n\}$.

In the RDF data model, any object from one triple can be a subject of another triple. This feature has been considered in Definition 3.3.2. where an Element Set es can contain a list of subjects or objects. It is necessary to mention that the SWARM approach has not considered an Element Set es as a list of predicates. Because, for each particular Semantic Item, the predicate of each Pair (p, o) plays a significant role in displaying the concept of a behaviour.

Semantic Items can be generated from triples by using Algorithm 3.1. The algorithm takes all *Triples* as an input and then generates a set of Semantic Items *SI* as an output. It extracts the Pair of each triple t_i and stores it in pa' (Lines 5-6). For each si_i in SI set, if pa' is the same as si_i's pair, then the algorithm adds the subject of triple t_i to $si_i.es$ (Lines 7-10). When there is no existing Semantic Item that contains a particular Pair, the algorithm creates a new Semantic Item si_i' and adds it to SI (Lines 13-16). The algorithm finally returns the updated SI in Line 20.

Algorithm 3.1: Semantic Item Generation

```
1: INPUT: $Triples$
2: OUTPUT: $SI$
3: $SI \leftarrow \varnothing$;
4: $foundFlag \leftarrow false$;
5: **for** each triple $t_j \in Triples$ **do**
6:    $pa' \leftarrow$ the Pair of $t_j$;
7:    **for** each $si_i \in SI$ **do**
8:       **if** $pa' = si_i.pa$ **then**
9:          add the subject of $t_j$ to $si_i.es$;
10:         $foundFlag \leftarrow true$;
11:      **end if**
12:   **end for**
13:   **if** foundFlag = false **then**
14:      new Semantic Item $si_i'$;
15:      $si_i' \leftarrow t_j$;
16:      add $si_i'$ to $SI$;
17:   **end if**
18:   $foundFlag \leftarrow false$;
19: **end for**
20: return $SI$;
```

Example 1. Consider the triples shown in Table 3.1. Some of the Semantic Items generated by Algorithm 3.1 have been shown in Table 3.2. For example, the Element Set of si_2 contains two entities *John Lennon* and *George Harrison*. The Pair of si_2 is (*occupation, Songwriter*). si_2 indicates that (*occupation, Songwriter*) is a particular behaviour of *John Lennon* and *George Harrison*. A related point to consider is that in this example, Semantic Items are generated by using the $\{s_1, s_2, ..., s_n\}(p, o)$ structure, i.e., an Element Set es contains a list of subjects.

Table 3.2: Some examples of Semantic Items

	Semantic Items
si_1	{*John Lennon, George Harrison*}(*instrument, Guitar*)
si_2	{*John Lennon, George Harrison*}(*occupation, Songwriter*)
si_3	{*Jimmy Carter, Bill Clinton, George W. Bush*}(*office, President of the USA*)
si_4	{*Jimmy Carter, Bill Clinton}(party, Democratic*)

Common Behaviour Set Generation

As explained earlier, a Semantic Item shows a common behaviour (described in the Pair) taken by a group of entities in the Element Set. In the following, a new concept called Common Behaviour Set is defined to represent all common activities taken by similar groups of entities in the Element Sets.

Definition 3.3.3. (*Common Behaviour Set*). A Common Behaviour Set cbs contains a set of Semantic Items with similar Element Sets, i.e., $cbs = \{si_1, si_2, ..., si_n\} = (ES, PA)$, where $ES = \{si_1.es \cup si_2.es \cup ... \cup si_n.es\}$ and $PA = \{si_1.pa \cup si_2.pa... \cup si_n.pa\}$. Items can be aggregated into the same *cbs*, if the similarity degree of their Element Sets is greater than or equal to Similarity Threshold *SimTh*. The Similarity Degree (*SD*) of Element Sets can be calculated by using Equation 3.1:

$$SD(es_a, es_b, ..., es_m) = \frac{|es_a \cap es_b \cap ... \cap es_m|}{|es_a \cup es_b \cup ... \cup es_m|} \tag{3.1}$$

According to Definition 3.3.3., a *cbs* is a set of Semantic Items aggregated based on the similarity of entities in Element Sets. A cbs shows a group of common occurrence of some activities taken by entities in the Element Sets. Accordingly, Total Common Behaviour Set $TCBS = \{cbs_1, cbs_2, ..., cbs_n\}$ contains all Common Behaviour Sets.

Algorithm 3.2: Total Common Behaviour Set Generation

```
1: INPUT: $SI$, $SimTh$
2: OUTPUT: $TCBS$
3: $TCBS \leftarrow \emptyset$;
4: $foundFlag \leftarrow false$;
5: for each $si_i \in SI$ do
6:    for each $cbs_j \in TCBS$ do
7:       $set_a \leftarrow \bigcap_{si_i \in cbs_j} si_i.es$;
8:       $set_b \leftarrow \bigcup_{si_i \in cbs_j} si_i.es$;
9:       $SD \leftarrow |si_i.es \cap set_a| / |si_i.es \cup set_b|$;
10:      if $SD \geq SimTh$ then
11:         add $si_i$ to $cbs_j$;
12:         $foundFlag \leftarrow true$;
13:      end if
14:   end for
15:   if foundFlag =false then
16:      new Common Behaviour Set $cbs'_j$;
17:      $cbs'_j \leftarrow si_i$;
18:      add $cbs'_j$ to $TCBS$;
19:   end if
20:   $foundFlag \leftarrow false$;
21: end for
22: return $TCBS$;
```

A $TCBS$ can be generated by Algorithm 3.2. The algorithm firstly receives Semantic Items SI=$\{si_1, si_2, ..., si_n\}$ and Similarity Threshold *SimTh* as inputs and then returns $TCBS$ as an output. For each si_i in SI, the algorithm computes *SD* of Element Sets of si_i and stores the result in *SD* (Lines 5-9). If SD is greater than or equal to *SimTh*, then the algorithm adds si_i to cbs_j (Lines 10-12). If there is no existing Common Behaviour Set with a similar Element Set, then the algorithm creates a new cbs'_j and adds it to $TCBS$ (Lines 15-18). After all, the algorithm returns the updated $TCBS$ in Line 22.

Example 2. Table 3.3 shows the Common Behaviour Sets generated from Semantic Items in Table 3.2. When *SimTh* equals to 50%, cbs_1 and cbs_2 can be generated from si_1, si_2 and si_3, si_4 (refer to Table 3.2), respectively. For example, cbs_2 shows that more than 50% of entities in its Element Sets are common in two behaviours, i.e., (*office, President of the USA*) and (*party, Democratic*).

Table 3.3: Common Behaviour Sets

	Common Behaviour Sets
$cbs1$	{*John Lennon, George Harrison*}(*instrument, Guitar*) {*John Lennon, George Harrison*}(*occupation, Song-writer*)
$cbs2$	{*Jimmy Carter, Bill Clinton, George W. Bush*}(*office, President of the USA*) {*Jimmy Carter, Bill Clinton*}(*party, Democratic*)

3.2.2 Mining module

The association rule mining fundamentally concentrates on extracting frequent co-occurring associations among a collection of items in the transactions. In this regard, there is a need to have a notion of frequency to generate Semantic Association Rules from the SW data. As discussed before, each cbs is a unique set and reveals the common occurrence of some activities taken by subjects or objects in the Element Sets. Namely, it can be considered as a particular form of transaction. Under this motivation, the following discusses how to generate Semantic Association Rules from Common Behaviour Sets. In order to show the role of semantics in the generated rules, quality rule factors (Support, Confidence, and Lift) are also defined by using the instance-level and schema-level knowledge.

Semantic Association Rules Generation

Definition 3.3.4. (*Semantic Association Rule*). A Semantic Association Rule *r* is composed by two parts: pa_{ant} and pa_{con}. pa_{ant} is the Antecedent Pairs and pa_{con} is the Consequent Pairs. Given a cbs_j, pa_{ant} contains some of the Pairs of cbs_j, i.e., $\{pa_1, ..., pa_n\}$. pa_{con} contains the remaining Pairs of cbs_j, i.e., $\{pa_{n+1}, ..., pa_m\}$. Rule *r* holds a common Rule's Element Set res which is the union of Element Sets in the cbs_j, i.e., $\{es_1 \cup ... \cup es_m\}$. Each Element Set es_i is a set of instances, i.e., $es_i = \{ins_1, ins_2, ..., ins_k\}$. We indicate a rule *r* with the antecedent and consequent by the implication below:

$$res\text{: } pa_{ant} \Longrightarrow pa_{con}$$

where res is a common Rule's Element Set containing $\bigcup_{si_i \in cbs_j} si_i.es$, $pa_{ant} \cap pa_{con} = \varnothing$, and $pa_{ant}, pa_{con} \in cbs_j$.

Example 3. Table 3.4 shows two examples of rules generated from Common Behaviour Sets in Table 3.3. For example, rule r_1 contains a common Rule's Element Set res generated by union of Element Sets in cbs_1, i.e., {*John Lennon, George Harrison*}. The antecedent and consequent of r_1 share the Pairs (*instrument, Guitar*) and (*occupation, Songwriter*), respectively. Note that the order of antecedents and consequents of rules can be swapped.

Table 3.4: Some examples of Semantic Association Rules

	Semantic Association Rules
r_1	{*John Lennon*, *George Harrison*}: ($instrument$, *Guitar*) ⇒ ($occupation$, *Songwriter*)
r_2	{*Jimmy Carter*, *Bill Clinton*, *George W. Bush*}: ($office$, *President of the USA*) ⇒ ($party$, *Democratic*)

Quality factors for rules

This chapter proposes three quality factors including Support, Confidence, and Lift that have used knowledge not only at the instance-level but also at the schema-level. In traditional association rule mining, transactions normally record the behaviours of one type (i.e., one class) of actors, i.e., shopping customers. However, in the SW, instances usually belong to different types or classes in the ontologies. In the SWARM, instances in the Rule's Element Sets also reflect this feature of the SW data.

As represented in Figure 3.1, *George Harrison* belongs to the Musical Artist class and *John Lennon* is an instance of Person class. Regarding to the concept of hierarchy in the ontology, if the Musical Artist class is a subclass of Artist class and the Artist class is a subclass of Person class, then the instances of Musical Artist class belong to the Person class as well. However, *John Lennon*, as an instance of Person class, does not belong to the Musical Artist class. It is an obvious challenge for interpreting association rules in the context of the SW data, and it fundamentally depends on the structure of an ontology. It does not make sense to measure the quality of rules by only considering knowledge at the instance-level. This observation leads us to use *rdf:type* and *rdfs:subClassOf* properties at the schema-level to measure the quality of rules.

Figure 3.3 depicts three possible hierarchical structures of an ontology. As shown in Figure 3.3(a), if Class c_1 is a subclass of Class c_3 through a middle class, i,e., c_2 ($c_1 \subseteq c_3$), then the instance I_a belongs to the Class c_3 as well. However, in Figure 3.3(b), Classes c_1 and c_5 are not in the same hierarchy ($c_1 \nsubseteq c_5$). In the ontology, classes on the upper levels present more general descriptions to compare with lower level classes which provide more specific descriptions. In this case, if classes are not in the same hierarchy, we just consider the Lowest Level Class (LLC) for each member in a Rule's Element Set. For example, in Figure 3.3(b), c_1 and c_5 is the LLC for I_a and I_b. In Figure 3.3(c), I_a and I_b belong to c_2 when the SWARM considers the LLC, while I_c is an instance of c_9. Consider again members in the Rule's Element Set r_1 shown in Table 3.4. The LLC for both *George Harrison* and *John Lennon* is the Person class.

Definition 3.3.5. (*Support*). Consider a Semantic Association Rule *r* in the form of $res : pa_{ant} \Longrightarrow pa_{con}$. The Support $sup\ (r)$ can be calculated by using Equation 3.2:

$$sup(r) = \frac{|\bigcup_{ins_j \in c_i \wedge ins_j \in res_k \wedge ins_j.pa_{ant_k}} c_i|}{|\bigcup_{ins_j \in c_i \wedge ins_j \in res_k} c_i|} \tag{3.2}$$

where $ins_j \in c_i$ is an instance of Class c_i, $ins_j \in res_k$ is a member of Rule's Element

Set res_k, and $ins_j.pa_{ant_k}$ shows those instances that share pa_{ant_k} as the Pairs. In this case, the numerator is the total number of instances of Class c_i that shares pa_{ant_k} as the Pairs. The denominator of the fraction is the total number of instances of Class c_i.

Example 4. Regarding to three different schemas shown in Figure 3.3, the generated rules for a, b, and c are r_a={I_a, I_b}: $pa_{ant} \Rightarrow pa_{con}$, r_b ={I_a, I_b}: $pa_{ant} \Rightarrow pa_{con}$, and r_c ={I_a, I_b, I_c}: $pa_{ant} \Rightarrow pa_{con}$, respectively. According to Equation 3.2, the Support values are:

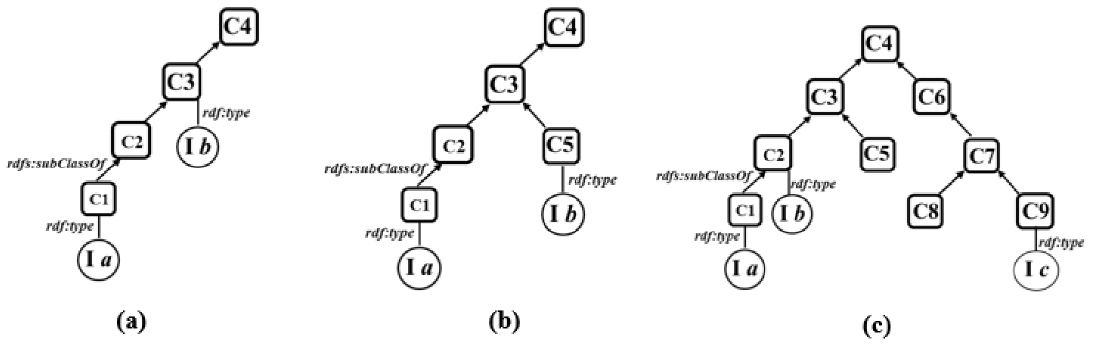

Figure 3.3: Different hierarchical structures of an ontology

$$sup(r_a) = \frac{|c_3 \cap pa_{ant}|}{|c_3|}$$
$$sup(r_b) = \frac{|(c_1 \cup c_5) \cap pa_{ant}|}{|c_1 \cup c_5|}$$
$$sup(r_c) = \frac{|(c_2 \cup c_9) \cap pa_{ant}|}{|c_2 \cup c_9|}$$

Example 5. The Support value of rule r_1 presented in Table 3.4 can be calculated by the following fraction.

$$sup(r_1) = \frac{|Person \cap instrument\ Guitar|}{|Person|}$$

The numerator of the Support fraction indicates the total number of instances of Person class that share (*instrument, Guitar*) as a Pair. As shown in Figure 3.1, there are only two instances which share (*instrument, Guitar*) as a Pair. The denominator of the fraction also shows the total number of instances of Person class which is six in this example (*sup*=0.33).

Definition 3.3.6. (*Confidence*). Consider the Semantic Association Rule *r* in the form of $res : pa_{ant} \Longrightarrow pa_{con}$. The Confidence $conf\ (r)$ can be computed by using Equation 3.3:

$$conf(r) = \frac{|\bigcup_{ins_j \in c_i \wedge ins_j \in res_k \wedge ins_j.pa_{ant_k} \wedge ins_j.pa_{con_k}} c_i|}{|\bigcup_{ins_j \in c_i \wedge ins_j \in res_k \wedge ins_j.pa_{ant_k}} c_i|} \tag{3.3}$$

where $ins_j \in c_i$ is an instance of Class c_i, $ins_j \in res_k$ is a member of Rule's Element Set res_k, and $ins_j.pa_{ant_k}$ and $ins_j.pa_{con_k}$ show those instances that share pa_{ant_k} and pa_{con_k} as the Pairs, respectively. In this case, the numerator is the total number of instances of Class c_i that shares pa_{ant} and pa_{con_k} as the Pairs. The denominator is the total number of instances of Class c_i that share pa_{ant_k} as the Pairs.

Example 6. The numerator of the Confidence fraction in rule r_1 shows the total number of instances in the Person class that contains (*instrument, Guitar*) and (*occupation, Songwriter*) as the Pairs. The denominator of the fraction also is the total number of instances that have been assigned to the Person class along with (*instrument, Guitar*) as a Pair (*conf*=1.0). The rule shows that most of the time people who play Guitar, they probably work as Songwriters.

$$conf(r_1) = \frac{|Person \cap instrument\ Guitar \cap occupation\ Songwriter|}{|Person \cap instrument\ Guitar|}$$

Definition 3.3.7. (*Lift*). Consider the Semantic Association Rule *r* in the form of $res : pa_{ant} \Longrightarrow pa_{con}$. $lift\ (r)$ can be measured by using Equation 3.4:

$$lift(r) = \frac{a}{b \times c} \tag{3.4}$$

where

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

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

$$c = \left| \bigcup_{ins_j \in c_i \wedge ins_j \in res_k \wedge ins_j.pa_{con_k}} c_i \right|$$

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

3.3 Experiments and analysis

In order to prove the usefulness of the SWARM, three main experiments are designed as follows. In the first experiment, the SWARM has been tested over the DrugBank dataset. The ontological structure of DrugBank dataset is simple. Therefore, the DBpedia dataset[2] has been used in the second experiment. The DBpedia is a very comprehensive triple store which provides a detailed ontology. The SWARM has been tested over six different classes of DBpedia ontology. In the third experiment, the SWARM has been compared with Mining Configurations (Abedjan & Naumann, 2013a, 2014) to demonstrate its usefulness.

Note that in the following experiments, the SWARM mines Semantic Association Rules by using the $\{s_1, s_2, ..., s_n\}(p, o)$ structure.

[2]https://wiki.dbpedia.org/services-resources/datasets/data-set-38/downloads-38

3.3.1 Experiment 1: Semantic Association Rules from DrugBank dataset

Experimental set-up. The DrugBank[3] dataset is a Bioinformatics and Cheminformatics resource which contains drug data information[4]. It contains 19,693 subjects, 276,142 objects, and 119 distinct predicates. The total number of triples in this dataset is 517,023.

Experimental results. The SWARM is tested over this dataset to check how readable the generated rules are. Two examples of discovered rules have been shown in Table 3.5. For example, Rule r_1 shows that the approved drugs with defined tablet oral dosage probably affect human and mammalian organisms.

Table 3.5: Examples of Semantic Association Rules from DrugBank (*SimTh=80%*)

Rule	Association Rule	sup.	conf.
r_1	{*owl:Thing*}: (*dosageForm*, *tabletOral*), (*drugType*, *Approved*) ⇒ (*AffectedOrganism*, *Humans and other mammals*))	0.33	1.0
r_2	{*owl:Thing*}: (*goClassificationFunction*, *electron transport (Transporter)*), (*goClassificationProcess*, *oxidoreductase activity (Refractivity)*) ⇒ (*goClassificationProcess*, *generation of precursor metabolites and energy*)	0.3	1.0

Table 3.6 represents the average value of Support, Confidence, and Lift of the generated rules by applying different Similarity thresholds. The total number of strong Semantic Association Rules generated from this dataset has also been shown in Figure 3.4.

Figure 3.5 depicts the general hierarchical structure of the DrugBank dataset. As seen, the granularity of DrugBank ontology [5] is limited and there is no specific hierarchical structure for the instance-level data. Note that in the figure, D_i, S_i, P_i, and O_m

[3]https://code.google.com/archive/p/fbench/
[4]https://www.drugbank.ca/documentation
[5]sadiframework.org/ontologies/LODD/drugbank.owl

stand for Drug name, Subject, Predicate and Object, respectively. Regarding to this simplicity, we selected another complex and comprehensive dataset called DBpedia for testing the SWARM approach.

Table 3.6: The quality measures of generated rules form DrugBank dataset

Min. SimTh%	Avg. sup.	Avg. conf.	Avg. lift
80	0.31	0.83	4.1
70	0.29	0.81	3.89
60	0.28	0.78	3.72

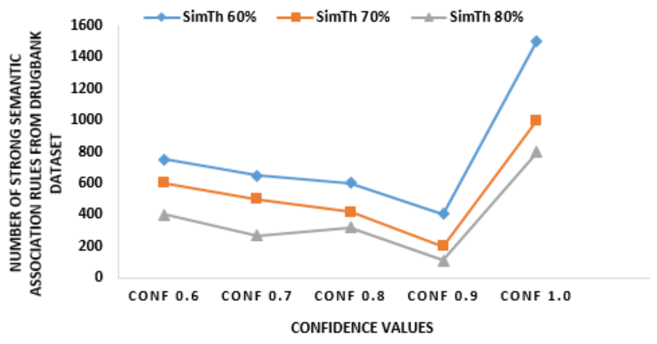

Figure 3.4: Number of strong rules with different minimum Similarity Threshold from DrugBank dataset

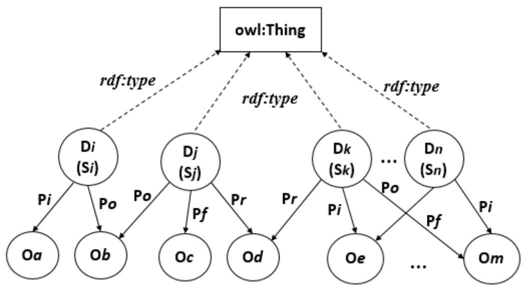

Figure 3.5: DrugBank hierarchical structure

3.3.2 Experiment 2: Semantic Association Rules from DBpedia dataset

Experimental set-up. The DBpedia project has been known as one of the most famous decentralized Linked Data efforts. Currently, the DBpedia project has released more than twelve comprehensive versions of its datasets along with the DBpedia ontology that is a very granular schema.

As a proof of concept, the SWARM has been tested over DBpedia (3.8). The DBpedia datasets usually provide two main files for data miners including *Ontology Infobox Properties* and *Ontology Infobox Types*. The *Ontology Infobox Properties* shares instance-level data, while the *Ontology Infobox Types* contains triples in the form of (*subject*, *rdf:type*, *ClassName*). $ClassName$ declares the name of classes for each subject in the DBpedia ontology. For example, $Anton\ Drexler$ belongs to the Politician, Person, and Agent classes in the ontology. In the following experiments, *Ontology Infobox Types* has been filtered based on 6 classes including Person, Organization, Place, Work, Event, and Species. In the DBpedia ontology, each of the above classes contains 26, 15, 10, 11, 9, and 3 main subclasses, respectively. Note that each main subclass refers to the first generation of leaf nodes produced by the respective class. Each subclass also treats as a class and may contain several subclasses. All these relations have been considered for testing the SWARM approach. By using triples filtered from *Ontology Infobox Types*, about 300,000 triples of *Ontology Infobox Properties* have been extracted (about 50,000 triples for each particular class). Some triples which share literal values (i.e., numbers and strings) have been removed from the subset dataset. Literal values such as *Birthdate* information are less interesting in the rule mining process. For each sample dataset, three separate experiments have been conducted by applying 60%, 70%, and 80% minimum Similarity Thresholds *SimTh*. The following results show that the SWARM can generate semantically-enriched rules from RDF data.

As a proof of concept, all members of Rule Element Sets are mentioned in the following tables. Instances in the DBpedia can be found on the Wikipedia Website, as DBpedia is a project aiming to exploit structured content from information created as part of the Wikipedia project.

Experimental results. The following tables show some interesting rules along with Support, Confidence and Lift values. Table 3.7 represents some Semantic Association Rules of Person class generated by *SimTh=60%*. For example, Rule r_1 shows that Scientists who are known for Natural selection theory are probably awarded the Copley and Royal Medals. Note that in this table, at least 60% of members of Element Sets satisfy the rules. Rule r_6 illustrates that Politicians who are residents of Amsterdam and works in the Labour Party of the Netherlands are probably residents of the Netherlands. Rule r_{11} shows that those Saints who are venerated in Lutheranism, which is a major branch of Protestant Christianity, are more likely to be venerated in the Anglican Communion. Rule r_{12} also indicates that people who were Kings of Macedon probably had Ancient Greek religion.

Table 3.8 shows some rules generated by $SimTh$ = 80%. Based on the DBpedia ontology, there exist some inconsistent patterns. Rule r_5 shows that some members of British Royal family who were born in Ribeira Palace and whose parents are Luisa of Guzman and John IV of Portugal, they have been probably buried in the Royal Pantheon of the House of Braganza. Although the members of Rule's Element Set r_5 satisfy the rule, none of them belongs to the British Royal family. More precisely, they are members of the Portuguese Royal family. Rules r_6 and r_7 also suffer from the same issue as r_5 does. In the DBpedia ontology, all royalties belong to the British Royalty and Polish King classes. The ontology does not define any other classes. Such inconsistencies between ontology definitions and the underlying data cause ambiguous interpretations. In the case of the DBpedia project, revising existing class definitions could be helpful to obtain a better understanding of data.

Some interesting Rules generated from Organization, Place, Work, and Event classes have been shown in Tables 3.9-3.12. Table 3.13 shows some rules of the Species class generated by $SimTh$ = 80%. For example, Rule r_2 says that Mammals of Balaenoptera category probably belong to Rorqual family. Rule r_3 of Table 3.10 is an interesting reflection of structure (c) in Figure 3.3 when instances are not located in the same hierarchy. The rule says that Mount Baker and Mount St. Helens are both volcanoes, while Half Dome is a well-known rock surface.

Table 3.14 shows the average value of Support, Confidence, and Lift of the generated rules with different Similarity Thresholds. Based on the results, the average support of rules generated from the Species class is much higher than other classes. It is directly related to the number of instances in this class. In the SWARM approach, the denominator of support fraction contains the total number of instances of some particular classes. The DBpedia ontology just defines three main subclasses for the Species class including Archaea, Bacteria, and Eukaryote. As a matter of fact, the lower number of instances causes a higher support value.

Figures 3.6-3.11 represent the number of strong Semantic Association Rules with different minimum Similarity Thresholds. The *SimTh* has a direct effect on the number of generated rules. Obviously, by increasing the *SimTh*, the number of generated rules will be decreased.

The conducted experiments show that the generated rules tend to have a low support rate. The intuition behind this result is that the denominator of support fraction usually contains the total number of instances of some classes. The real-world RDF-style KBs contain a huge number of instances that causes a low support rate generated by the SWARM. The approach also generates a lot of rules with confidence 1.0. As seen in the Figures 3.6-3.11, the number of rules with $0.9 < conf <= 1.0$ has increased sharply. The intuition behind this fact is because of filtering mechanism for generating common

Behaviour sets.

Table 3.7: Examples of Semantic Association Rules discovered from Person class (*SimTh=60%*)

Rule	Rule's Element Set	Semantic Association Rule	sup.	conf.	lift
r_1	{*Alfred Russel Wallace, Charles Darwin, Andrew Wiles, Robert Bunsen*}	{*Scientist*}: (*knownFor*, *Natural selection*) ⇒ (*award*, *Copley Medal*), (*award*, *Royal Medal*)	0.01	1.0	4.06
r_2	{*Neil Peart, Alex Lifeson, Ozzy Osbourne , Buck Dharma, Jim Morrison, Eric Bloom, Ice-T*}	{*Artist*}: (*genre*, *Heavy metal music*) ⇒ (*genre*, *Hard rock*)	0.02	0.90	3.42
r_3	{*John Major, Tony Blair, William Ewart Gladstone*}	{*PrimeMinister*}: (*religion*, *Church of England*) ⇒ (*orderInOffice*, *Prime Minister of the United Kingdom*)	0.01	0.75	38.51
r_4	{*Frank D. White, Joe Purcell*}	{*Governer*}: (*successor*, *Bill Clinton*) ⇒ (*orderInOffice*, *Governor of Arkansas*) (*office*, *Arkansas Attorney General*)	0.02	0.67	58.46
r_5	{*Mihail Savov, Nikola Ivanov, Radko Dimitriev, Ivan Fichev*}	{*Person*}: (*battle*, *Battle of Pirot*), (*battle*, *Battle of Kresna Gorge*), (*country*, *Bulgaria*)⇒ (*militaryBranch*, *Bulgarian Land Forces*)	0.03	1.0	22.82
r_6	{*Lodewijk Asscher, Eberhard van der Laan*}	{*Politician*}: (*residence*, *Amsterdam*), (*party*, *Labour Party (Netherlands)*) ⇒ (*residence*, *Netherlands*)	0.04	1.0	10.26
r_7	{*William H. Rupertus, Roy Geiger, Raymond A. Spruance*}	{*Person*}: (*militaryBranch*, *United States Marine Corps*) ⇒ (*award*, *Navy Cross*)	0.02	1.0	46.68

r_8	{*Benjamin Disraeli*, *William Ewart Gladstone*, *Alexander Mackenzie*,*John A. Macdonald*}	{*PrimeMinister*}: ($orderInOffice$, *Leader of the Opposition*) ⇒ ($monarch$, *Queen Victoria*)	0.02	0.75	37.01
r_9	{*Liam Fox*, *William Hague*, *Oliver Letwin*}	{*Person*}: ($office$, *Leader of the Conservative Party*) ⇒ ($office$, *Shadow Foreign Secretary*)	0.08	1.0	50.33
r_{10}	{*Wu Bangguo*, *Wen Jiabao*, *Jia Qinglin*,*Hu Jintao*}	{*Person*}: ($party$, *Communist Party of China*) ⇒ ($office$, *National People's Congress*)	0.03	1.0	11.50
r_{11}	{*Augustine of Hippo*, *Saint Titus*, *Bernard of Clairvaux*, *Athanasius of Alexandria*}	{*Saint*}: ($veneratedIn$, *Lutheranism*) ⇒ ($veneratedIn$, *Anglican Communion*)	0.04	0.87	24.28
r_{12}	{*Amyntas I of Macedon*, *Alcetas I of Macedon*, *Alexander I of Macedon*, *Alcetas II of Macedon*, *Perdiccas II of Macedon*}	{*Person*}: ($title$, *King of Macedon*) ⇒ ($religion$, *Religion in ancient Greece*)	0.02	1.0	41.07

Table 3.8: Examples of Semantic Association Rules discovered from Person class (*SimTh=80%*)

Rule	**Rule's Element Set**	**Semantic Association Rule**	**sup.**	**conf.**	**lift**
r_1	{*Bill Gates*, *Mary Maxwell Gates*, *Adam West*}	{*Person*}: ($birthPlace$, *Seattle*)⇒ ($birthPlace$, *Washington (state)*)	0.02	0.91	17.77

r_2	{*Torch (rapper)*, *Toni L*, *Christopher Franke*	{*Artist*}: ($genre$, *German hip hop*) ($associatedBand$, *Advanced Chemistry*) ⇒ ($hometown$, *Germany*)	0.06	1.0	12.68
r_3	{*David Hume*, *John Stuart Mill*}	{*Philosopher*}: ($influenced$, *Bertrand Russell*) ⇒ ($philosophicalSchool$, *Utilitarianism*)	0.1	0.67	11.7
r_4	{*Ivan Turgenev*, *Guy de Maupassant*, *Sheridan Le Fanu*}	{*Writer*}: ($influenced$, *Henry James*) ⇒ ($genre$, *Realism (arts)*)	0.09	0.87	5.08
r_5	{*Afonso VI of Portugal*, *Peter II of Portugal*}	{**BritishRoyalty**}: ($birthPlace$, *Ribeira Palace*), ($parent$, *Luisa of Guzman*), ($parent$, *John IV of Portugal*) ⇒ ($restingPlace$, *Royal Pantheon of the House of Braganza*)	0.04	1.0	4.92
r_6	{*Alfonso V of Aragon*, *John II of Aragon*}	{**BritishRoyalty**}: ($parent$, *Ferdinand I of Aragon*), ($birthPlace$, *Medina del Campo*) ⇒ ($parent$, *Eleanor of Alburquerque*)	0.04	1.0	4.3
r_7	{*John I Albert*, *Casimir IV Jagiellon*}	{**BritishRoyalty**}: (*birthPlace*, *Kraków*) ⇒ ($restingPlace$, *Wawel Cathedral*), ($deathPlace$, *Poland*)	0.06	1.0	4.0

Table 3.9: Examples of Semantic Association Rules discovered from Organization class (*SimTh=80%*)

Rule	Rule's Element Set	Semantic Association Rule	sup.	conf.	lift

r_1	{*Communist Party of China*, *Communist Party of the Soviet Union*, *Polish United Workers' Party*, *Socialist Unity Party of Germany*}	{*PoliticalParty*}: (*ideology*, *Marxism-Leninism*)⇒ (*ideology*, *Communism*)	0.02	0.83	13.83
r_2	{*University of Sudbury*, *University of Waterloo*, *University of Ottawa*, *York University*}	{*University*}: (*state*, *Ontario*) ⇒ (*affiliation*, *Council of Ontario Universities*)	0.11	1.0	4.8
r_3	{*Holden*, *Opel*, *Hummer*, *Chevrolet*, *Cadillac*, *Cadillac*}	{*Company*}: (*owner*, *General Motors*) ⇒ (*owningCompany*, *General Motors*)	0.03	1.0	8.7
r_4	{*YMCK*, *Dragon Ash*, *Mr. Children*, *King Giddra* }	{*Band*}: (*hometown*, *Japan*) ⇒ (*hometown*, *Tokyo*)	0.15	0.62	6.33
r_5	{*Royal Australian Air Force*, *Royal Australian Navy*, *Royal Air Force*, *Blue Angels*}	{*MilitaryUnit*}: (*aircraftFighter*, *Boeing F/A-18E/F Super Hornet*) ⇒ (*garrison*, *Canberra*)	0.09	1.0	8.06

Table 3.10: Examples of Semantic Association Rules discovered from Place class (*SimTh=80%*)

Rule	**Rule's Element Set**	**Semantic Association Rule**	**sup.**	**conf.**	**lift**

r_1	{*Liverpool*, *Coventry*, *Manchester*}	{*City*}: ($governmentType$, *Metropolitan borough*)⇒ ($isPartOf$, *North West England*)	0.15	0.67	7.95
r_2	{*Siderno*, *Locri*, *Reggio Calabria*}	{*Settlement*}: ($region$, *Calabria*) ⇒ ($province$, *Province of Reggio Calabria*)	0.02	1.0	3.5
r_3	{*Mount Baker*, *Mount St. Helens*, *Half Dome*}	{*Volcano* ∪ *Mountain*}: ($locatedInArea$, *United States*) ⇒ ($mountainRange$, *Cascade Range*)	0.12	0.67	8.24
r_4	{*Snowy Mountains*, *Great Dividing Range*}	{*Mountain*}: ($state$, *New South Wales*) ⇒ ($highestPlace$, *Mount Kosciuszko*), ($state$, *Victoria* (*Australia*))	0.04	0.85	4.07
r_5	{*Madeira River*, *Amazon River*	{*River*}: ($timeZone$, *Time in Brazil*) ⇒ ($country$, *Brazil*)	0.1	1.0	9.40

Table 3.11: Examples of Semantic Association Rules discovered from Work class (*SimTh=80%*)

Rule	**Rule's Element Set**	**Semantic Association Rule**	**sup.**	**conf.**	**lift**
r_1	{*Choke (novel)*, *Survivor (Chuck Palahniuk novel)*, *Catch-22*}	{*Book*}: ($literaryGenre$, *Black comedy*)⇒ ($author$, *Chuck Palahniuk*)	0.16	0.67	6.34
r_2	{*Apache HTTP Server*, *OpenOffice*, *Xerces*, *Xalan*}	{*Software*}: ($developer$, *Apache Software Foundation*) ⇒ ($license$, *Apache License*)	0.02	1.0	5.9

r_3	{*Puzzle Bobble*, *Bubble Bobble*, *Space Invaders*, *Zero Wing*, *Darius II (video game)*}	{*VideoGame*}: (*developer*, *Taito Corporation*) ⇒ (*publisher*, *Taito Corporation*)	0.03	1.0	2.54
r_4	{*The Big Lebowski*, *Miller's Crossing*}	{*Film*}: (*director*, *Coen brothers*) ⇒ (*producer*, *Coen brothers*)	0.24	0.85	7.31
r_5	{*Led Zeppelin II*, *Led Zeppelin III*, *Led Zeppelin IV*, *Physical Graffiti*, *Houses of the Holy*}	{*Album*}: (*genre*, *Blue rock*)⇒ (*artist*, *Led Zeppelin*)	0.12	1.0	3.26

Table 3.12: Examples of Semantic Association Rules discovered from Event class (*SimTh=80%*)

Rule	**Rule's Element Set**	**Semantic Association Rule**	**sup.**	**conf.**	**lift**
r_1	{*Irish Civil War*, *Irish War of Independence*, *Easter Rising*}	{*MilitaryConflict*}: (*place*, *Irish Free State*)⇒ (*combatant*, *Irish National Army*), (*combatant*, *Irish Republican Army*)	0.07	1.0	13.80
r_2	{*Summerfest*, *Bonnaroo Music Festival*}	{*MusicFestival*}: (*genre*, *Americana (music)*) ⇒ (*genre*, *Rhythm and blues*), (*genre*, *Contemporary classical music*)	0.04	1.0	3.3
r_3	{*Live Aid*, *1966 FIFA World Cup Final*}	{*Event*}: (*location*, *London*) ⇒ (*location*, *United Kingdom*)	0.02	1.0	3.88

Table 3.13: Examples of Semantic Association Rules discovered from Species class (*SimTh=80%*)

Rule	Rule's Element Set	Semantic Association Rule	sup.	conf.	lift
r_1	{*Bacillus thuringiensis*, *Bacillus cereus*, *Bacillus*}	{*Bacteria*}: ($genus$, *Bacillus*)⇒ ($family$, *Bacillaceae*)	0.2	1.0	21.65
r_2	{*Blue whale*, *Humpback whale*, *Fin whale*, *Bryde's whale*, *Minke whale*, *Antarctic minke whale* }	{*Mammal*}: ($genus$, *Balaenoptera*) ⇒ ($family$, *Rorqual*)	0.61	0.99	2.14
r_3	{*Western Grebe*, *Clark's Grebe*, *Pacific Loon*}	{*Bird*}: ($genus$, *Aechmophorus*) ⇒ ($binomialAuthority$, *George Newbold Lawrence*)	0.33	1.0	3.74
r_4	{*Puffball*, *Geastrales*, *Gymnosporangium*, *Armillaria*, *Cantharellaceae*, *Russulales*, *Chanterelle*, *Boletales*, *Shiitake*, *Homobasidiomycetidae*}	{*Fungus*}: ($division$, *Basidiomycota*) ⇒ ($class$, *Agaricomycetes*)	0.51	1.0	3.68
r_5	{*Rat snake*, *Squamata*, *Phymaturus*, *Lizard*}	{*Reptile*}: ($order$, *Lepidosauria*),($class$, *Diapsid*) ⇒ ($class$, *Lepidosauromorpha*)	0.52	0.96	21.67
r_6	{*Zebrafish*, *Koi*, *Cyprinus*, *Acrossocheilus*, *Cobitidae*, *Quillback*, *Mahseer*, *Tanichthys*}	{*Fish*}: ($family$, *Cyprinidae*) ⇒ ($order$, *Cypriniformes*)	0.36	1.0	3.23

r_7	{*Hallucigenia*, *Aysheaia*, *Microdictyon*}	{*Animal*}: ($order$, *Scleronychophora*) ⇒ ($class$, *Xenusiid*)	0.09	1.0	15.07
r_8	{*Hellbender*, *Axolotl*, *Toad*, *Tiger Salamander*, *Palmate Newt*}	{*Amphibian*}: ($order$, *Salamander*) ⇒ ($class$, *Amphibian*)	0.31	0.75	3.20
r_9	{*Branchiopoda*, *Crayfish*, *Crustacean*, *Portunus*, *Barnacle*, *Krill*, *Zarigani*, *Astacus*, *Mystacocarida*, *Bylgia*}	{*Crustacean*}: ($phylum$, *Crustacean*) ⇒ ($class$, *Malacostraca*)	0.33	0.75	3.0
r_{10}	{*Scorpion*, *Pseudoscorpion*, *Palpigradi*, *Mygalomorphae*, *Latrodectus*, *Uloboridae*, *Ixodidae*, *Kimura*, *Theridiidae*, *Sicariidae*, *Opiliones*, *Acari*}	{*Arachnid*}: ($class$, *Arachnid*) ⇒ ($order$, *Spider*)	0.63	0.72	3.9
r_{11}	{*Nanoarchaeum equitans*, *Nanoarchaeota*, *Halobacteria*}	{*Archaea*}: ($class$, *Archaea*) ⇒ ($domain$, *Archaea*)	0.50	1.0	2.0
r_{12}	{*Spirotrich*, *Heterotrich*, *Plagiopylida*, *Hymenostome*, *Apicomplexa*, *Plasmodium*, *Dinoflagellate*, *Peniculid*}	{*Eukaryote*}: ($phylum$, *Ciliate*) ⇒ ($phylum$, *Alveolate*)	0.35	1.0	4.18

Table 3.14: The quality measures of generated rules form different classes of DBpedia ontology

Class	**Min. SimTh** %	**Avg. sup.**	**Avg. conf.**	**Avg. lift**
Person	80	0.02	0.97	29.11
	70	0.018	0.93	24.22
	60	0.012	0.61	20.15
Organization	80	0.03	0.95	4.59
	70	0.02	0.94	4.31
	60	0.01	0.90	3.80
Place	80	0.037	0.97	9.70
	70	0.031	0.95	8.8
	60	0.013	0.89	7.74
Work	80	0.02	0.97	5.68
	70	0.015	0.94	5.50
	60	0.011	0.87	4.1
Event	80	0.029	0.97	3.38
	70	0.023	0.93	3.05
	60	0.01	0.91	2.59
Species	80	0.36	0.95	3.98
	70	0.32	0.92	3.72
	60	0.3	0.85	3.12

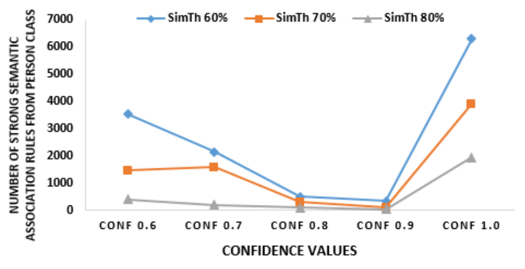

Figure 3.6: Number of strong rules in different minimum Similarity Thresholds from Person class

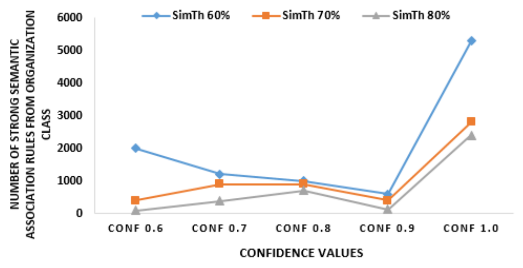

Figure 3.7: Number of strong rules in different minimum Similarity Thresholds from Organization class

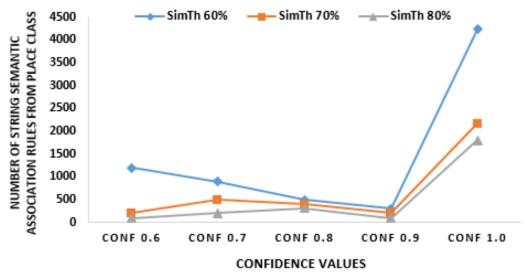

Figure 3.8: Number of strong rules in different minimum Similarity Thresholds from Place class

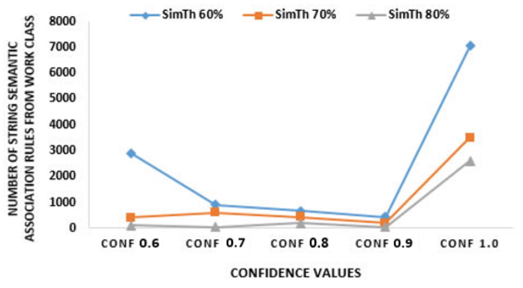

Figure 3.9: Number of strong rules in different minimum Similarity Thresholds from Work class

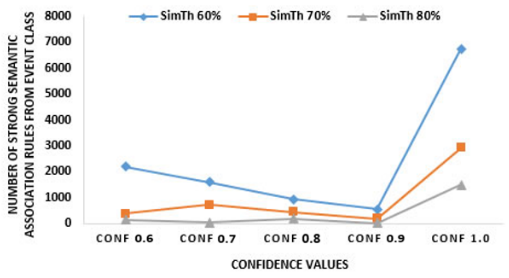

Figure 3.10: Number of strong rules in different minimum Similarity Thresholds from Event class

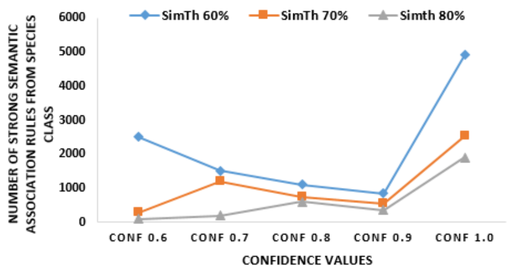

Figure 3.11: Number of strong rules in different minimum Similarity Thresholds from Species class

3.3.3 Experiment 3: SWARM vs. Mining Configurations

In the following, the semantics of rules generated by the SWARM has been compared with Mining Configurations (Abedjan & Naumann, 2013a, 2014) and the importance of using relations and schema-level knowledge has been highlighted for interpreting rules. Then, the SWARM and Mining Configurations are compared based on other quality metrics. The SWARM, as well as Mining Configurations, generates association rules from RDF-style KBs. The SWARM mines association rules by utilizing knowledge at the instance-level and schema-level. However, the Mining Configurations method considers only the instance-level data and it mines rule without considering the ontological properties in the ontology. Note that Abedjan and Naumann (2013a) tested Mining Configurations over the DBpedia dataset (3.6).

To provide a fair comparison, the Mining Configurations has been developed and tested over the same sample dataset from DBpedia (3.8). We have carried out several experiments using Mining Configurations on instances of the Person class. In these experiments, the *rdf:type* and *rdfs:subClassOf* properties are disregarded from the dataset. In other words, all instances are considered as members of the Person category. The following demonstrates that the SWARM outperforms the Mining Configurations by attaching more semantics to the rules.

Semantics of generated rules. Mining Configurations is developed by using the concept of association rule mining. It performs a depth-first search using the FP-Growth algorithm (Han et al., 2000) to discover frequent patterns from RDF data. The configurations have been shown in Table 3.15.

Table 3.15: Six configurations of context and target

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

Abedjan and Naumann (2013a) claimed that mining subjects in the context of predicates results in the clustering of entities. For example, *George Washington⇒Lyndon B. Johnson* can be classified as presidents since *George Washington* and *Lyndon B. Johnson* are both presidents of the United States of America. Although Mining Configurations helps to express some clusters, it does not guarantee that all entities in a rule belong to the same class. The results of running Config. 3 on the sample dataset shows that it is not possible to identify a precise cluster for entities without considering the ontological properties in the ontology. Consider the following rule obtained by the Mining Configurations:

$$\textit{Thomas Aquinas} \Rightarrow \textit{Plato}, \textit{Duns Scotus}$$

In order to interpret the above rule generated by Config. 3 shown in Table 3.15, we can conclude that *Thomas Aquinas*, *Plato* and *Duns Scotus* are subjects that are correlated by some common predicates. The information regarding such a correlation has been kept hidden in the rule. It is interesting to attach predicates to express more semantics in the rules. Apart from this limitation, *Thomas Aquinas* is a member of Saint class, while *Plato* and *Duns Scotus* are instances of Philosopher class in the DBpedia ontology. Here the question is that what does this cluster mean?

Table 3.16: Semantic Association Rules generated by the SWARM (*SimTh=80%*)

Rule	**Rule's Element Set**	**Semantic Association Rule**	**sup.**	**conf.**	**lift**
r_1	{*Thomas Aquinas*, *Duns Scotus*, *Averroes*}	{*Saint* ∪ *Philosopher*}: ($influncedBy$, *Aristotle*)⇒ ($influncedBy$, *Plato*)	0.08	0.91	8.77
r_2	{*Friedrich Nietzsche*, *Max Stirner*, *Roland Barthes*, *Judith Butler*}	{*Philosopher*}: ($philosophicalSchool$, *Post structuralism*)⇒ ($philosophicalSchool$, *Post modernism*)	0.1	0.75	43.5

In comparison with Mining Configurations, the SWARM represents the relationships among *Thomas Aquinas*, *Plato* and *Duns Scotus* as shown in Table 3.16. Rule r_1 demonstrates that a group of saints and philosophers who influenced by Aristotle are probably influenced by Plato. In the Element Set of Rule 1, *Thomas Aquinas* belongs to the Saint class, while *Duns Scotus* and *Averroes* are instances of Philosopher class. Rule 2 also shows another interesting pattern related to 20th Century movements in the philosophy and literary criticism. As seen, the SWARM not only considers knowledge at the instance-level but the schema-level.

The mining results obtained by Config. 4 (Mining Objects) is very similar to Config. 3. However, this configuration still suffers from the same issue that Config. 3 does. Similarly to the above explanation, the following rule achieved by Config. 4 indicates the necessity of using *rdf:type* and *rdfs:subClassOf* properties.

Aristotle, *Augustine of Hippo* ⇒ *Anselm of Canterbury*, *gottfried wilhelm leibniz*

In the above rule, *Aristotle* and *gottfried wilhelm leibniz* belong to the Philosopher class, while *Augustine of Hippo* and *Anselm of Canterbury* have been assigned to the Saint class in the DBpedia ontology.

The mining results from Config. 2 and Config. 5 are also not enriched enough. Abedjan and Naumann (2013a) mentioned that mining predicates in the context of subjects (refer to Config. 1 in Table 3.15) could be used for the schema discovery. For example, an association rule *associatedBand, instrument⇒ associtedMusicalArtist* shows a schema for musicians. The results of mining association rules by Config. 1 on our dataset shows that in some cases it is not possible to detect a certain schema for a rule. Because predicates of a particular rule might be common among different classes in the ontology. Consider the following rule discovered by Mining Configurations.

$$influencedBy \Rightarrow influenced, birthPlace, deathPlace$$

The schema is not recognizable from the above rule. It is because *influencedBy* is a common predicate among Philosopher, Saint, and Writer classes in the DBpedia ontology. Config. 6 also mines predicates in the context of objects has the same issue as Config. 1. Such inconsistencies have been solved in the SWARM by utilizing *rdf:type* and *rdfs:subClassOf* properties.

Comparison of other quality metrics. Although the strategy of mining rules in the SWARM differs from Mining Configurations, the results represented in Table 3.17 show that the SWARM obtains a higher average of confidence and lift values. Table 3.17 also shows that the average number of generated transactions by SWARM is much higher than Mining Configurations.

Table 3.17: Mining Configurations vs. SWARM (*0.6≤Min conf.≤0.8*)

Approach	**Avg. sup.**	**Avg. conf.**	**Avg. lift**	**Avg. ♯ Transactions**
Mining Configurations (*0.02≤Min Sup≤0.2*)	0.21	0.49	3.44	376
SWARM (*0.6≤SimTh≤0.8*)	0.02	0.83	29.11	1137

It is interesting to know that the average of support value obtained by Mining Configurations is higher than the SWARM. The Mining configurations applies traditional support measure to evaluate the quality of rules. In such a case, the support value of *X* of Rule *r:X⇒Y* with respect to the number of transactions is defined as the proportion of transactions which contains X. While, the SWARM approach considers *rdf:type* and *rdfs:subClassOf* properties to compute the quality of rules. Thus, the result of using these relations causes a lower support average in the SWARM.

3.4 Discussion

The following discusses the extensibility of the SWARM. It is worth mentioning that the SWARM has the potential of using *rdfs:subPropertyOf* and *rdf:Property* which are OWL properties. Similarly to the ontological hierarchy for entities, properties also have their own hierarchy in the ontology. The ontological property *rdfs:subPropertyOf* is an instance of *rdf:Property* that is used to define that one property is a sub-property of another, i.e., *rdfs:subPropertyOf* behaves similarly to *rdfs:subClassOf*. Consider again the Semantic Association Rule r_6 in Table 3.7 which is shown as follows:

$$\{\textit{Politician}\}\text{: } (residence, \textit{Amsterdam}), (party, \textit{Labour Party (Netherlands)}) \Rightarrow (residence, \textit{Netherlands})$$

In the DBpedia ontology, *residence*[6] and *party*[7] are sub-properties of *hasLocation* and *isMemberOf*, respectively. Namely, *hasLocation* and *isMemberOf* properties are super-properties of residence and party. Since *rdfs:subPropertyOf* property is transitive, it is possible to replace *residence* or *party* with *hasLocation* or *isMemberOf* properties. Therefore, the rule r_6 can be transformed into the following form:

[6]http://dbpedia.org/ontology/residence
[7]http://dbpedia.org/ontology/party

$$\{\textit{Politician}\}\colon (hasLocation, \textit{Amsterdam}), (isMemberOf, \textit{Labour Party (Netherlands)}) \Rightarrow (hasLocation, \textit{Netherlands})$$

Replacing *rdfs:subPropertyOf* properties with the corresponding super-properties has a direct influence on generating rules with more general descriptions. Conversely, replacing super-properties with sub-properties transform rules into more descriptive patterns. However, this task needs special attention. In the DBpedia property hierarchy, *location* is super-property of two sub-properties called *locationCity* and *locationCountry*. Replacing the *location* with *locationCity* or *LocationCountry* without considering the *rdfs:range* of objects may cause rules to transform in a wrong fashion. Because, the *rdfs:range* of *locationCity* is City class while the *rdfs:range* of *locationCountry* is Country class in the DBpedia ontology. To avoid such misinterpretations, considering the *rdfs:range* of objects is essential. A related point to consider is that in the RDF triple (subject, predicate, object), the *rdfs:range* of the predicate denotes the class type of the object in the ontology.

As discussed above, ontological properties can be useful for mining hidden patterns from the SW data. Powerful ontology languages such as OWL have been developed to formulate complex semantics relations. Using ontological properties strongly depend on the goal of research. Consider the functional property *owl:FunctionalProperty* on the topic of reasoning in ontologies. This property defines just one particular value *y* for each instance *x*. For example, a person is born in exactly one city, etc. If one of these values is observed, then observable models can prevent other values from being asserted.

3.5 Summary

The main contribution of this chapter is to reveal common behavioural patterns from RDF data by considering instance-level and schema-level knowledge. The SWARM is a novel approach that attaches semantics to the rules by utilizing relations and knowledge encoded at the schema-level. The approach can automatically mine Semantic Association Rules from RDF-style KBs. The SWARM improves the Mining Configurations (Abedjan & Naumann, 2013a) by considering the relations and schema-level knowledge in the ontology. The SWARM takes advantage of *rdf:type* and *rdfs:subClassOf* properties to generate semantically-enriched rules. The chapter also explains how class information of entities has been used to calculate Support, Confidence, and Lift values. Hence, the qualified rules reveal common behavioural patterns among different types of entities. The discovered rules can also be enriched by using other properties (e.g., *rdfs:subPropertyOf* and *rdf:Property*). Initial experiments conducted on RDF-style KBs (DrugBank dataset and some classes of DBpedia dataset (3.8)) show the effectiveness of the SWARM. The results indicate that the SWARM outperforms the Mining Configurations regarding the semantics of discovered rules. It is important to note that the SWARM is applicable to any type of RDF-style KB.

Based on the results achieved by the SWARM, the SW data suffer from lack of correctness and consistency between entities at the instance-level and classes in the ontology (refer to Table 3.8). Such inconsistencies between ontological definitions and underlying data cause vague interpretations. In this respect, Chapter 4 proposes an approach to deal with such conflicts.

The work in this chapter has been published in (Barati, Bai & Liu, 2016, 2017).

Chapter 4

An Entropy-Based Class Assignment Detection Approach For RDF Data

The SW technologies provide a flexible approach for publishing the SW data over the LOD cloud. However, RDF-style KBs usually suffer from a considerable amount of faulty facts which are known as SW data quality issues. Due to the complexity of relationships, the SW data quality issues are continuously growing. In this respect, revisiting the consistency between instance-level and schema-level can purify the KBs.

The experimental results in Chapter 3 revealed that there exist some inconsistent patterns in the DBpedia dataset. Consider again Rule r_7 in Table 3.8. The rule shows that *John I Albert*, king of Poland, has been incorrectly assigned to the BritishRoyalty class instead of the PolishKing class defined in the DBpedia ontology. Based on this observation, this chapter focuses on a SW data quality issue called Incorrect Class Assignment (ICA). The occurrence of this issue is inevitable since the process of distributing the SW data is handled by users. Furthermore, this chapter introduces an approach called Class Assignment Detector (CAD) to analyse the correctness and incorrectness of relationships between instances and classes in the ontology.

The rest of this chapter is organized as follows. Section 4.1 provides a motivating example to formally define the ICA problem. Section 4.2 describes the details of the CAD approach. The experimental results obtained by the CAD are explained in Section 4.3. Finally, Section 4.4 reviews the whole chapter along with contributions of the CAD approach.

4.1 Motivation

Consider the following association rule obtained by the SWARM from DBpedia dataset (3.8):

{*John I Albert, Casimir IV Jagiellon*}: {**BritishRoyalty**}: (*birthPlace, Kraków*) ⇒ (*restingPlace, Wawel Cathedral*), (*deathPlace, Poland*)

The rule indicates that some members of British royal family, i.e., *John I Albert and Casimir IV Jagiellon*, who were born in the *Kraków* buried and died in the *Wawel Cathedral* and *Poland*, respectively. The DBpedia ontology defines a Royal class with two subclasses of BritishRoyalty and PolishKing for all royalties. There exist some instances that are incorrectly assigned to unrelated classes in the ontology. For example, *John I Albert*, king of Poland, has been assigned to BritishRoyalty class instead of PolishKing class defined in the DBpedia ontology. This example can be identified as an incorrect assignment between an instance and the class in the ontology.

The above problem has been illustrated in Figure 4.1. Given an ontology with its associated instances: all instances that have been correctly assigned to classes in the ontology are called CA. The data quality issue can be defined as ICA problem where at least one instance has been incorrectly assigned to class A instead of class B. Under this motivation, this chapter proposes the CAD approach to tackle the ICA problem.

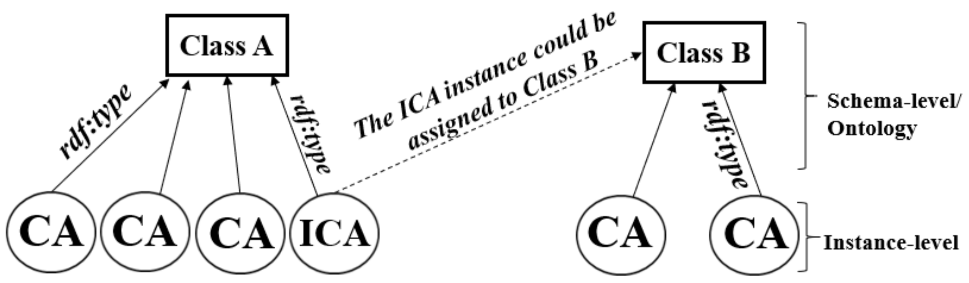

Figure 4.1: Modelling the ICA problem

4.2 Class Assignment Detector (CAD) approach

The CAD framework contains two main modules: (I) Class Features Extraction module, and (II) Instance-Class Relationship Analysis module. In the Class Features Extraction module, features of classes in the existing ontology are extracted through analysing instances. The output of the Class Features Extraction module is used in the Instance-Class Relationship Analysis module to evaluate the correctness and incorrectness of relationships between instances and the classes.

4.2.1 Class Features Extraction module

Generally, a class is a category of things having some common features that make those things distinct from others. To detect the features of classes, the initial step is to analyse RDF triples to discover their common features. The following explains the idea behind mining common features from RDF triples. Then, the process of extracting features of classes is described in details.

Identifying common features from RDF triples

As previously mentioned, the SW is built based on two main levels including instance-level which takes RDF triples and schema-level/ontology which defines the semantics for RDF triples. The assertion of an RDF triple, i.e., (subject, predicate, object), shows a meaningful relationship between a subject and an object prepared by a predicate. As explained before, a triple can be considered as the description of one particular feature of instances (i.e., subject and object).

Table 4.1: Some RDF triples from the DBpedia dataset

Triples	**Subject**	**Predicate**	**Object**
t_1	*Queen Victoria*	*birthPlace*	*United Kingdom*
t_2	*Queen Victoria*	*spouse*	*Albert, Prince Consort*
t_3	*Edward VII*	*birthPlace*	*United Kingdom*
t_4	*Princess Beatrice of the UK*	*birthPlace*	*United Kingdom*
t_5	*Princess Beatrice of the UK*	*parent*	*Queen Victoria*
t_6	*Princess Louise, Duchess of Argyll*	*birthPlace*	*United Kingdom*
t_7	*Princess Louise, Duchess of Argyll*	*restingPlace*	*United Kingdom*
t_8	*United Kingdom*	*leaderTitle*	*"Monarch"@en*
t_9	*John I Albert*	*birthPlace*	*Poland*
t_{10}	*John I Albert*	*parent*	*Casimir IV Jagiellon*
t_{11}	*John I Albert*	*deathPlace*	*Poland*
t_{12}	*Casimir III*	*birthPlace*	*Poland*
t_{13}	*Casimir III*	*predecessor*	*Władysław I the Elbow high*
t_{14}	*Przemysł II*	*birthPlace*	*Poland*
t_{15}	*Przemysł II*	*successor*	*Wenceslaus II of Bohemia*
t_{16}	*Sigismund I the Old*	*birthPlace*	*Poland*
t_{17}	*Poland*	*longName*	*"Republic of Poland "@en*
t_{18}	*Raul Solnado*	*deathPlace*	*Portugal*
t_{19}	*Lucília do Carmo*	*deathPlace*	*Portugal*
t_{20}	*Portugal*	*longName*	*"Portuguese Republic"@en*
t_{21}	*France*	*leaderTitle*	*"President"@en*
t_{22}	*Zeca Afonso*	*deathPlace*	*Portugal*
t_{23}	*Florbela Espanca*	*deathPlace*	*Portugal*

Consider Triple t_{18} (*Raul Solnado, deathPlace, Portugal*) shown in Table 4.1. If we consider *Raul Solnado* as an instance, then the other two elements in the triple (i.e., *deathPlace Portugal*) can be identified as a particular feature of *Raul Solnado*. Now consider Triple t_{22} (*Zeca Afonso, deathplace, Portugal*) in Table 4.1. The subjects in t_{18} and t_{22}, i.e., *Raul Solnado* and *Zeca Afonso*, have a common feature, i.e., (*deathPlace*

Portugal). In other words, being dead in the *Portugal* is the common feature taken by *Raul Solnado* and *Zeca Afonso*. To analyse the common features shared by different instances, we give the following definitions.

Definition 4.3.1. (*Group Feature*). Given RDF triples, a Group Feature gf_i is a 2-tuple, i.e., gf_i= (g_i, f_i). g_i is a Group that contains a list of subjects or objects, i.e., $\{s_1, s_2, ..., s_n\}$ or $\{o_1, o_2, ..., o_n\}$. f_i is a Feature of g_i. Corresponding with the content in g_i, f_i contains a combination of predicate-object or predicate-subject, i.e., (p, o) or (p, s).

Definition 4.3.2. (*Common Feature*). Given a Group Feature gf_i=(g_i, f_i), the Feature f_i is a Common Feature cf for g_i, if the number of instances in the g_i is greater than or equal to Minimum Instance Number (*MinIN*).

According to Definition 4.3.2., triples in a triple store can have a set of Common Features i.e., CFs = $\{cf_1, cf_2, ..., cf_n\}$. Note that, we have not defined a Group g as a list of predicates; because the predicate of each Feature indicates a particular role in expressing the concept of a feature.

Example 1. For triples in Table 4.1, if $MinIN$ = 4, the following three common features can be identified by (*birthPlace, United Kingdom*) shared by {*Queen Victoria*, *Edward VII*, *Princess Louise*, *Princess Beatrice*}, (*birthPlace, Poland*) shared by {*Casimir III, John I Albert, Sigismund I the Old, Przemysł II}*, and (*deathPlace, Portugal*) shared by {*Raul Solando, Zeca Afonso, Florbela Espanca, Lucília do Carmo*}.

Detecting features of classes

To tackle the ICA problem, CAD takes advantage of the information theory (Shannon, 1949) to analyse the uncertainty related to the correctness or incorrectness of relationships between instances and classes. As explained, a Common Feature shows a common behaviour of instances in a Group. The information gain allows us to measure which common features are more certain to be used as key features of classes to distinguish their instances.

The following introduces a measure based on entropy that calculates the uncertainty associated with the whole random space in the SW. Then, it explains how to compute information gained by common features.

Definition 4.3.3. (*Class Random Space Entropy*). Given an ontology, the Class Random Space S is a space built up from instances of different classes in the ontology. The entropy of *S* can be calculated by Equation 4.1:

$$Entropy(S) = -\sum_{i=1}^{N} p(c_i) \log_2 p(c_i) \tag{4.1}$$

where N is the total number of classes in the ontology and $p(c_i) = \frac{|Ins_{c_i}|}{|INS|}$ is the probability of Class c_i in S. $|Ins_{c_i}|$ is the total number of instances of Class c_i. $|INS|$ is the total number of instances in S.

In the SW, an instance can belong to multiple classes in the ontology. This fact increases the entropy (i.e., uncertainty) of Class Random Space measured by Equation 4.1, but it is obviously not a data quality issue. In this module, the target of CAD approach is to find features of classes. However, belonging to multiple classes can introduce extra uncertainty to the process of class features extraction. The following shows how the CAD approach handles extra uncertainty occurred by instances with multiple types.

In the information theory, more information can be obtained by a random space with lower entropy, and vice versa. In this scenario, a Common Feature can be shared by instances of different classes in the ontology. Therefore, the information gained from a Common Feature depends on the types of instances in its Group. Fewer types cause lower entropy and consequently more information gained by the Common Feature. By relying on the fact, the concepts of Common Feature Space and Common Feature Information are defined as follows.

Definition 4.3.4. (*Common Feature Space*). Given a Common Feature cf_k, RDF triples and its corresponding ontology, a Common Feature Space S_{cf_k} is a space built up from instances that share cf_k. The Entropy of S_{cf_k} can be computed by Equation 4.2:

$$Entropy(S_{cf_k}) = -\sum_{i=1}^{m_{S_{cf_k}}} p(c_{j.cf_k}) \log_2 p(c_{j.cf_k}) \tag{4.2}$$

where

$$m_{S_{cf_k}} = nc_t - \sum((\sum_{nbs_{i.cf_k}} \frac{1}{ns_{i_j}}) - min(\frac{1}{ns_{i_j}})) \tag{4.3}$$

$m_{S_{cf_k}}$ is the total number of classes in S_{cf_k}, nc_t is the number of belonging classes of instances with cf_k, ns_{i_j} is the number of supporting instances with cf_k of Class c_j, $nbs_{i.cf_k}$ is the number of belonging classes of an instance ins_i with cf_k. $p(c_{j.cf_k}) = \frac{|Ins_{c_j.cf_k}|}{|Ins_{cf_k}|}$ is the probability of class c_j in S_{cf_k}. $|Ins_{c_j.cf_k}|$ is the total number of instances of class c_j that share cf_k. $|Ins_{cf_k}|$ is the total number of instances that share cf_k.

Definition 4.3.5. (*Common Feature Information*). Given a Class Random Space *S* and a Common Feature cf_k, the information gained from cf_k is a normalized value measured by Equation 4.4:

$$Gain(S, cf_k) = \frac{Entropy(S) - \frac{|Ins_{cf_k}|}{|INS|} Entropy(S_{cf_k})}{Entropy(S)} \tag{4.4}$$

where *Entropy(S)*≠ 0 and 0≤ $Gain(S, cf_k)$ ≤1.

The Common Feature Space can be calculated by using Algorithm 4.1. The algorithm receives classes (*C*), instances (*Iset*), features of instances (*I.features*), and types of instances (*I.types*) as inputs. Then, it returns $Entropy(S_{cf_k})$ as an output. For each instance ins_i that shares Common Feature cf_k, the algorithm records its belonging classes/types in $bs_{i.cf_k}$ (Lines 3-5). Then, it extracts the maximum number of supporting instances for classes in $bs_{i.cf_k}$ and stores the value in $min_{ns_{i_j}}$ (Line 6). For each Class c_j in $bs_{i.cf_k}$, if the number of supporting instances with cf_k of c_j is lower than $min_{ns_{i_j}}$, then the algorithm updates $min_{ns_{i_j}}$ and records c_j as a Class of ins_i (c_{ins_i}) (Lines 7-10). It continues this process to find out the minimum number of ns_{i_j} for ins_i and reduces extra uncertainty introduced by multiple types of ins_i which is also reflected in Equation 4.3. Then, the algorithm considers c_{ins_i} as a related class (i.e., class type) of ins_i (Line 13). Finally, the algorithm measures $Entropy(S_{cf_k})$ by using related classes in Line 16.

Algorithm 4.1: Common Feature Space Calculation

```
 1: INPUT: $C = \{c_1, ..., c_j, ..., c_n\}$, $Iset = \{ins_1, ..., ins_i, ..., ins_n\}$,
    $I.features = \{ins_1.features, ..., ins_i.features, ..., ins_n.features\}$,
    $I.types = \{ins_1.types, ..., ins_i.types, ..., ins_n.types\}$, $cf_k$
 2: OUTPUT: $Entropy(S_{cf_k})$
 3: for each $ins_i \in Iset$ do
 4:    if $ins_i$ has $cf_k$ then
 5:       $bs_{i.cf_k} \leftarrow$ belonging classes of $ins_i$ ;
 6:       $min_{ns_{i_j}} \leftarrow$ maximum of $ns_{i_j}$ among all Classes in $bs_{i.cf_k}$ ;
 7:       for each $c_j \in bs_{i.cf_k}$ do
 8:          if $ns_{i_j} < min_{ns_{i_j}}$ then
 9:             $min_{ns_{i_j}} \leftarrow$ add $ns_{i_j}$ ;
10:             $c_{ins_i} \leftarrow$ add $c_j$ ;
11:          end if
12:       end for
13:       $related\ c \leftarrow$ add $c_{ins_i}$ ;
14:    end if
15: end for
16: $Entropy(S_{cf_k}) \leftarrow$ calculate Entropy with $related\ c$;
17: return $Entropy(S_{cf_k})$;
```

Example 2. Table 4.2 lists the belonging classes of the triples in Table 4.1. Table 4.2 also shows the corresponding Class Random Space created by instance-level and schema-level knowledge. In case that BritishRoyalty, PolishKing, Actor, MusicalArtist, Poet, Singer, and Country classes contain 6, 6, 1, 1, 1, 1, and 4 instances, the total number of instance-level data in this Class Random Space is equal to 20. Using Equation 4.1, the entropy of this Class Random Space equals to 2.36.

Table 4.2: An example of a Class Random Space

Schema-level	**Instance-level data**
BritishRoyalty Class (*rdf:type BritishRoyalty*)	*Queen Victoria, Edward VII, Princess Beatrice of UK, Princess Louise Duchess of Argyll, Albert Prince Consort, John I Albert*
PolishKing Class (*rdf:type PolishKing*)	*Casimir III, Przemysł II, Casimir IV Jagiellon, Władysław I the Elbow high, Wenceslaus II of Bohemia, Sigismund I the Old*
Country Class (*rdf:type Country*)	*United Kingdom, Poland, Portugal, France*
Actor Class (*rdf:type Actor*)	*Raul Solando*
MusicalArtist Class (*rdf:type MusicalArtist*)	*Zeca Afonso*
Singer Class (*rdf:type Singer*)	*Lucília do Carmo*
Poet Class (*rdf:type Poet*)	*Florbela Espanca*

Example 3. The common features of (*birthPlace, United Kingdom*), (*birthPlace, Poland*), and (*deathPlace, Portugal*) can create three common feature spaces. Table 4.3 shows the details of each Common Feature Space. For example, the Common Feature Space created by (*birthplace, United Kingdom*) contains instances of BritishRoyalty class. This Common Feature has been shared by 4 instances of BritishRoyalty class. The entropy of Common Feature Space $S_{(birthPlace,\, United\ Kingdom)}$ shown in Table 4.3 is equal to 0 since all instances belong to the same class. In this case, *Gain* (*S*, (*birthPlace, United Kingdom*)) obtains maximum value that is equal to 1. This value indicates that

there exists no uncertainty in the random space of (*birthPlace*, *United Kingdom*) since all instances have the same type that is *rdf:type* BritishRoyalty. Since Common Feature (*birthPlace, Poland*) is shared by 1 instance of BritishRoyalty class and 3 instances of PolishKing class, the entropy of $S_{(birthPlace,\ Poland)}$ is equal to 0.807 and the value of *Gain* (*S*, (*birthPlace, Poland*)) is about 0.93.

Table 4.3: Examples of common feature spaces

Common features	**Schema-level**	**Instance-level data**
(*birthPlace, United Kingdom*)	BritishRoyalty	*Queen Victoria, Edward VII, Princess Beatrice of UK, Princess Louise Duchess of Argyll*
(*birthPlace, Poland*)	BritishRoyalty	*John I Albert*
	PolishKing	*Casimir III, Przemysł II, Sigismund I the Old*
(*deathPlace, Portugal*)	Actor	*Raul Solnado*
	MusicalArtist	*Zeca Afonso*
	Singer	*Lucília do Carmo*
	Poet	*Florbela Espanca*

On one side, a Common Feature can be shared by instances of different classes. On the other side, instances might share more than one Common Feature. The CAD has also evaluated the information gained from combinations of common features. For example, in Figure 4.2, Common Feature cf_1 is shared by instance ins_a with *rdf:type D*, and ins_b, ins_i, ins_n with *rdf:type A*, and ins_j with *rdf:type B*. Common Feature cf_2 is also shared by ins_b, ins_i, ins_n with *rdf:type A*, and ins_k with *rdf:type C*, and ins_m with *rdf:type E*. The combination of (cf_1,cf_2) gains more information to compare with each cf_1 and cf_2. Because the Random Space of (cf_1,cf_2) contains lower entropy and fewer types (i.e., types *A* and *B*) to compare with cf_1 and cf_2 that contain instances with {*rdf:type A*, *rdf:type B*, *rdf:type D*} and {*rdf:type A*, *rdf:type C*, *rdf:type E*}, respectively. Based on this motivation, the concept of Virtual Common Feature is defined as follows.

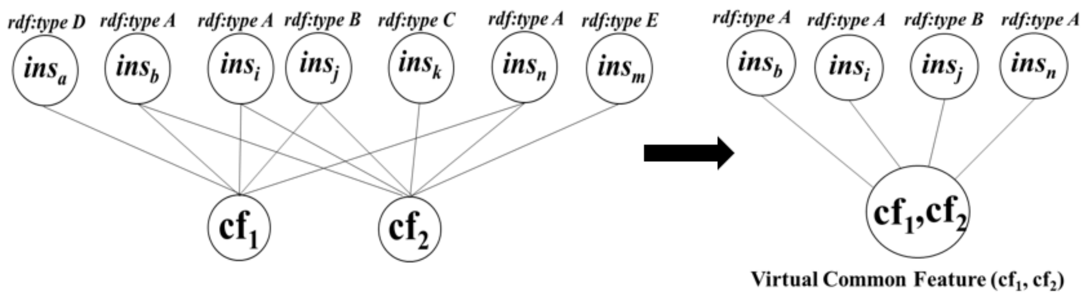

Figure 4.2: An example of a Virtual Common Feature

Definition 4.3.6. (*Virtual Common Feature*). A Virtual Common Feature vcf is a combination of n ($n \geq 2$) common features where the number of instances that share vcf is greater than or equal to *MinIN*.

A Virtual Common Feature vcf shows that there exist a number of instances that share more than one Common Feature. In this regard, the information gained from a Virtual Common Feature can be computed by Equations 4.4. We just need to replace $|Ins_{cf_k}|$ and Entropy (S_{cf_k}) with $|Ins_{vcf_k}|$ and Entropy (VS_{vcf_k}). The $|Ins_{vcf_k}|$ is the total number of instances that share vcf_k and Entropy (VS_{vcf_k}) is the Entropy of a Virtual Common Feature Space.

Given a Common Feature cf_k and a Virtual Common Feature vcf_k, the more information indicates lower entropy (i.e., lower uncertainty) in the random spaces generated by cf_k and vcf_k. This fact reveals that most instances that have shared cf_k and vcf_k have the same type. Based on this fact, the concept of Class Feature is defined as follows.

Definition 4.3.7. (*Class Feature*). A Common Feature cf_k or a Virtual Common Feature vcf_k is a Class Feature for Class c_i, if the information gained by cf_k or vcf_k is greater than or equal to NormGain Thresholds (*NGTh*).

According to Definition 4.3.7., a class can have multiple class features including common features and virtual common features with information gained greater than or equal to *NGTh*, i.e., Class $c_i.features$={cf_1,cf_2,...,cf_k,...,vcf_n,vcf_{n+1},...}, where cf_k or vcf_n is a Class Feature for Class c_i if instances that share cf_k or vcf_n are members of Class c_i.

4.2.2 Instance-Class Relationship Analysis module

If we take an instance with belonging types/classes and its features, the goal of Instance-Class Relationship Analysis module is to evaluate the correctness (i.e., CA: if an instance shares its class features) and incorrectness (i.e., ICA: if an instance does not share its class features) of relationships between instances and classes in the existing Knowledge Base (KB). To this end, Algorithm 4.2 is proposed in this module. In the following, an *Undecidable* status is also considered for an instance when it is neither CA nor ICA. The status of an instance is Undecidable when it has not shared any Common Feature or the information gained by its common features or virtual common features is lower than *NGTh*. Algorithm 4.2 also has the capability of receiving and analysing new RDF instances as input data into the KB. In this respect, the algorithm takes advantage of batch processing which is a method for storing data into groups to allow sequential processing and reduce system overhead. In the algorithm, *InstanceBatch* refers to instances sharing common features and virtual common features. If the number of new instances in *BatchCounter* equals to the maximum allowed number of entries in *BatchMax*, then the algorithm analyses *InstanceBatch*. Algorithm 4.2 receives *MinIN*, *NGTh*, classes (*C*), features of classes (*C.features*), instances (*Iset*), features of instances (*I.features*), and types of instances (*I.types*) as inputs. Then, it returns *CA*, *ICA*, and *Undecidable* relationships of instances with the classes as outputs.

Algorithm 4.2: Instance-Class Relationship Analysis

```
1: INPUT: $MinIN$, $NGTh$, $C = \{c_1, ..., c_j, ..., c_n\}$,
   $C.features = \{c_1.features, ..., c_j.features, ..., c_n.features\}$,
   $Iset = \{ins_1, ..., ins_i, ..., ins_n\}$,
   $I.features = \{ins_1.features, ..., ins_i.features, ..., ins_n.features\}$,
   $I.types = \{ins_1.types, ..., ins_i.types, ..., ins_n.types\}$
2: OUTPUT: $CA$, $ICA$, $Undecidable$
3: for each $ins_i \in Iset$ do
4:   if $ins_i$ has one Common Feature then
5:     If $ins_i$ is not new, adds it to $Iset_{cf}$ ;
6:     If $ins_i$ is new, adds it to $InstanceBatch$;
7:   else if $ins_i$ has no Common Feature then
8:     $Undecidable \leftarrow ins_i$ is neither CA nor ICA;
9:   else
10:      $Feature_{ins_i} \leftarrow$ Common Features of $ins_i$;
11:      if $Feature_{ins_i} \geq MinIN$ then
12:        $vcf_{ins_i} = Feature_{ins_i}$;
13:        If $ins_i$ is not new, adds it to $Iset_{vcf}$ ;
14:        If $ins_i$ is new, adds it to $InstanceBatch$;
15:      else
16:        If $ins_i$ is not new, adds it to $Iset_{cf}$ ;
17:        If $ins_i$ is new, adds it to $InstanceBatch$;
18:      end if
19:    end if
20:  end for
21:  if $(|InstanceBatch| == BatchMax)$ then
22:    Update $Iset_{cf}$ and $Iset_{vcf}$ with $InstanceBatch$;
23:  end if
24:  for each $ins_i \in Iset_{cf}$ do
25:    if Information gained by each Common Feature of $ins_i \geq NGTh$ then
26:      if Common Feature of $ins_i \in \{c_j.features \mid c_j \in C\}$ then
27:        if $(ins_i.types$ contains $c_j.type)$ then
28:          $CA \leftarrow ins_i$ status ;
29:          If $ins_i$ is new, recalculate $Entropy(S)$ and $Gain(S, Common\,Feature\,of\,ins_i)$;
30:        else
31:          $ICA \leftarrow add\ \{ins_i, c_j.type\}$;
32:          Recalculate $Entropy(S)$ and $Gain(S, Common\,Feature\,of\,ins_i)$;
33:        end if
34:        $break$;
35:      end if
36:    else
37:      $Undecidable \leftarrow ins_i$ is neither CA nor ICA;
38:    end if
39:  end for
```

```
40: for each $ins_i \in Iset_{vcf}$ do
41:     if Information gained by $vcf_{ins_i} \geq NGTh$ then
42:         $classType \leftarrow Type\ vcf_{ins_i}$ ;
43:         if $(ins_i.types$ contains $classType)$ then
44:             $CA \leftarrow ins_i$ status;
45:             If $ins_i$ is new, recalculate $Entropy(S)$ and $Gain(S, vcf_{ins_i})$;
46:         else
47:             $ICA \leftarrow add\ \{ins_i, classType\}$;
48:             Recalculate $Entropy(S)$ and $Gain(S, vcf_{ins_i})$;
49:         end if
50:     else
51:         $Undecidable \leftarrow ins_i$ is neither CA nor ICA;
52:     end if
53: end for
54: return $CA$, $ICA$, $Undecidable$;
```

For each instance ins_i that shares a Common Feature, if it exists in the dataset, then the algorithm adds it to $Iset_{cf}$ (Lines 3-5). If ins_i has not shared any Common Feature, then the status of ins_i is *Undecidable* (Lines 7-8). If ins_i shares multiple common features, the algorithm stores all common features of ins_i in $Feature_{ins_i}$ to check the possibility of creating virtual common features (Line 10). If the number of instances that have shared $Feature_{ins_i}$ is greater than or equal to *MinIN*, then the algorithm creates virtual common features vcf_{ins_i} and records them in $Iset_{vcf}$ (Lines 11-13). Otherwise, the algorithm records ins_i with common features in $Iset_{cf}$ (Line 16). If ins_i is a new instance for the KB, then the algorithm adds it to *InstanceBatch* (Lines 14 and 17). If the algorithm starts batch processing, then $Iset_{cf}$ and $Iset_{vcf}$ will be updated by *InstanceBatch* by considering common features and virtual common features (Lines 21-22).

For each ins_i in $Iset_{cf}$, if the information gained by each Common Feature of ins_i is greater than or equal to *NGTh*, then the algorithm checks the Common Feature of ins_i in the features of all classes (Lines 24-26). If the Common Feature of ins_i exists in the features of Class c_j, then the algorithm compares the types of ins_i with the *type* of Class c_j (Line 27). The status of ins_i is CA, if ins_i types contains in Class c_j type (Line 28). If ins_i types do not contain in Class c_j type, then the algorithm considers

ICA status for ins_i and assigns it to Class c_j (Line 31). Since an instance with ICA status has been added to Class c_j, the algorithm recalculates the entropy of whole Class Random Space and updates the information gained by Class Feature of c_j that in fact, it is the Common Feature shared by ins_i (Line 32). In case of adding a new instance that has CA status, the algorithm measures *Entropy*(S) and the information gained by Class Feature of c_j (Line 29). Note that the status of ins_i is Undecidable if the information gained by its common features is lower than *NGTh* (Line 37).

For each ins_i in $Iset_{vcf}$, if the information gained by vcf_{ins_i} has exceeded *NGTh*, then the algorithm extracts *classType* of instances that have shared vcf_{ins_i} (Lines 40-42). If ins_i type is the same as extracted *classType*, then the status of ins_i is CA (Lines 43-44). Otherwise, ins_i has ICA status and the algorithm assigns it to *classType* (Line 47). The status of ins_i is *Undecidable*, if the information gained by vcf_{ins_i} does not exceed *NGTh* (Line 51).

4.3 Experiments and analysis

The following has been structured based on two main experiments. The first experiment aims to assess the capability of the CAD approach in discovering features of classes. The second experiment has been designed to evaluate the accuracy of the CAD approach in detecting the correctness and incorrectness of relationships between instances and classes. Note that in the following experiments, the CAD mines common features by using $\{s_1, s_2, ..., s_n\}$(p,o) structure where (p,o) is a Common Feature for a Group of subjects.

4.3.1 Experiment 1: Class Feature extraction from DBpedia dataset

Experimental set-up. The following experiments have been conducted over DBpedia dataset (3.8). As explained in Chapter 3, DBpedia datasets usually provide two main files including *Ontology Infobox Properties* that publishes RDF triples and *Ontology Infobox Types* that provides schema-level knowledge for the triples. In the following experiments, we filtered the *Ontology Infobox Types* based on 10 classes including Hotel, Food, ArchitecturalStructure, Museum, Beverage, Event, Holiday, BritishRoyalty, Person, and Writer. By using triples filtered from *Ontology Infobox Types*, about 4800 triples have been extracted from *Ontology Infobox Properties* for the experiments. Each of classes including Food, Hotel, and Holiday approximately contain 750 instances. ArchitecturalStructure, Museum, Beverage, Event, BritishRoyalty, Person, and Writer classes also have about 350 instances for each one.

Experimental results. The goal of the first experiment is to test the ability of the CAD in extracting features of classes by applying NormGain Thresholds $NGTh \geq 0.85$ and Minimum Instance Number $MinIN \geq 15$.

Table 4.4: Some details of Hotel, Food, Beverage, Holiday, and BritishRoyalty classes ($NGTh \geq 0.85$, $MinIN \geq 15$)

Class	**class features**	**Gain**	**Instances**
Hotel	(*location,UK*)	0.88	{*Draycott Hotel, 22 Jermyn Street, Bedford Hotel (Brighton), Blakes Hotel, ...*}
	(*location,Laos*)	0.87	{*Amantaka, Lao Plaza Hotel, Royal Dokmaideng Hotel, Settha Palace Hotel, ...*}
Food	(*origin,India*)	0.92	{*Tilkut, Barfi, Chutneg, Kaju katli, Neer dosa, Roti canai, Ramja, Samber, Angoori, Luchi, Chomchom, Ariselu, Paneer, Putto, Rasam, ...*}

	{(*origin,Indonesia*), (*origin,Malaysia*)}	0.95	{*Otak-otak, Coconut rice, Laksa, Mie goreng, Ayam goreng, Asam pedas, ...*}
	(*origin,Philippines*)	0.91	{*Taho, Sigsig, Asocena, batchoy, Baye baye, Bibingka, Curacha, Gulaman, Sapin Sapin, Tinola, Mechado, Kalamay, kaldereta, Halo halo, ...*}
Beverage	(*origin,Scotland*)	0.93	{*Barr Cola, Buchanan's, Chivas Regal, Cutty Sark Whisky, Irn Bru, John Dewar & Sons, Johnnie Walker, Red Kola, Royal Salute Whisky, Valadivar Vodka, ...*}
	(*origin,Peru*)	0.89	{*Concordia beverage, Inca Kola, Isaac Kola, Kola Inglesa, Kola Real, Triple Kola, Pulp juice, Fruti Kola, Pilsen Callao, ...*}
	(*origin, Australia*)	0.87	{*Bombra Vodka, Mother(energy drink), Count Cola, Cooranbong Vodka, Dot AU Vodka, LA ICE Cola, Solo Australian Soft Drink, Schweppes Cola, ...*}
Holiday	(*country, Hindu*)	0.9	{*Balipratipada, Nag Panchami, Ayudha Puja, Hanuman Jayanti, Akshaya Tritiya, Chhath, Nuakhai, Ratha Saptami, Ganesh Chaturthi, Prabodhini Ekadashi, Gadhimai Festival, Krishna Janmashtami, Kumbh Mela, Meha Shivaratri, Diwali, ...*}
	(*country, Judaism*)	0.86	{*Shavuot, Shemini Atzeret, Simchat Torah, Sukkot, Rosh Hashanah, Hanukkah, Purim, Birkat Hachama, Yom Kippur, ...*}

	(*country, USA*)	0.85	{*Blasphemy Day, Obama Day, Leif Erikson Day, Malcolm X Day, Juneteenth, Kwanzaa, Veterans Day, Super Saturday, ...*}
British-Royalty	(*country, United Kingdom*)	0.91	{*Edward, the Black Prince, Henry I of England, Edward III of England, Harold Godwinson, Kenneth MacAlpin, Anne Boleyn, Edward the Confessor, Henry VII of England, Mary, Princess Royal and Princess of Orange, James II of England, Charles II of England, Jane Seymour, ...*}

Table 4.4 shows some features of Hotel, Food, Beverage, and BritishRoyalty classes along with information gained by each Class Feature. The table also provides some instances that have shared features of these classes. For example, (*origin,India*) is a Class Feature in the Food class that has gained information greater than *NGTh*≥ 0.85. As shown in Table 4.4, {(*origin,Indonesia*),(*origin,Malaysia*)} is an example of a Virtual Common Feature identified as a Class Feature for the Food class. This Class Feature reveals there exist some common food between Indonesia and Malaysia.

4.3.2 Experiment 2: Accuracy of CAD approach

Experimental set-up. The goal of the second experiment is to check the accuracy of CAD approach in analysing the correctness and incorrectness of relationships between instances and classes in the dataset. Generally, the accuracy of a system is a degree of closeness between a measured value and a true value. In this scenario, a measured value refers to the number of correctly assigned (i.e., CA) and incorrectly assigned (i.e, ICA) instances detected by the CAD approach. While a true value indicates the predefined number of CA and ICA instances of a class. To this end, 700 instances with *rdf:type* Hotel are considered as a true value for CA instances. While, 50 hotel instances are intentionally and incorrectly considered as instances of Food class (i.e., they have *rdf:type* Food). These 50 instances show a true value for ICA instances. To measure the accuracy of CAD approach, two measurements called $Accuracy_{CA}$ and $Accuracy_{ICA}$ are defined as follows.

Definition 4.4.8. ($Accuracy_{CA}$). Given a class, the accuracy of the CAD in detecting correctly assigned instances can be computed by Equation 4.5:

$$Accuracy_{CA} = \frac{|ins_{CA} \cap INS_{CA}|}{|INS_{CA}|} \tag{4.5}$$

where ins_{CA} is the number of correctly assigned instances detected by the CAD and INS_{CA} is a true value for the predefined number of CA instances in the class.

Definition 4.4.9. ($Accuracy_{ICA}$). Given a class, the accuracy of the CAD in detecting incorrectly assigned instances can be measured by Equation 4.6:

$$Accuracy_{ICA} = \frac{|ins_{ICA} \cap INS_{ICA}|}{|INS_{ICA}|} \tag{4.6}$$

where ins_{ICA} is the number of incorrectly assigned instances detected by the CAD and INS_{ICA} is a true value for the predefined number of ICA instances in the class.

Experimental results. According to the dataset and instances, Figure 4.3 shows that the accuracy of CAD approach in detecting CA instances is positively grown by increasing *NGTh*. As explained previously, a Common Feature or a Virtual Common Feature is a Class Feature of Class c_i, if the information gained by the Common Feature or Virtual Common Feature is greater than or equal to *NGTh*. This fact indicates lower entropy (i.e., higher certainty) in the random space generated by the Common Feature or Virtual Common Feature. Based on Algorithm 4.2, the status of an instance is Undecidable if the information gained by its common features and virtual common features is lower than *NGTh*. Subsequently, Algorithm 4.2 disregards ins_i, if it has an Undecidable status. Similarly to what explained above, Figure 4.4 also represents that the accuracy of the CAD in detecting ICA instances is raised by increasing *NGTh*. Figures 4.5 and 4.6 show that the number of detected CA and ICA instances is grown by increasing *NGTh*.

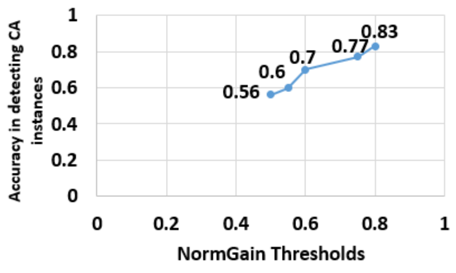

Figure 4.3: Accuracy of the CAD in detecting CA instances in different *NGTh*

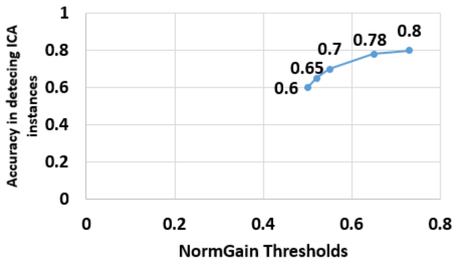

Figure 4.4: Accuracy of the CAD in detecting ICA instances in different *NGTh*

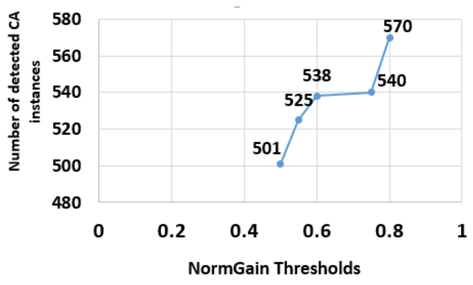

Figure 4.5: Number of detected CA instances in different *NGTh*

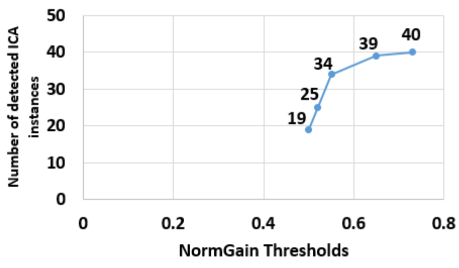

Figure 4.6: Number of detected ICA instances in different *NGTh*

It is important to mention that Undecidable statuses directly impact on the accuracy of CAD approach as shown in the above figures. Consider the process of analysing common features in Algorithm 4.2. Given an instance ins_i, if the feature shared by ins_i is not a Common Feature, then the status of ins_i is Undecidable. For example, *White House* is an instance with the *rdf:type* Hotel that shares (*location, Herm*) as a particular feature. More precisely, (*location, Herm*) is a single feature that is only shared by *White House*. Based on the concept of Common Feature and *MinIN* thresholds, (*location, Herm*) cannot be considered as a Common Feature since it is shared by only one instance. Furthermore, if *White House* has been incorrectly assigned to the Food class, Algorithm 4.2 ignores *White House* as an ICA instance since (*location, Herm*) is not a Common Feature. Accordingly, Algorithm 4.2 disregards some instances if the features shared by them are not common features in the dataset.

It is also important to note that the impact of *MinIN* on the $Accuracy_{CA}$ and $Accuracy_{ICA}$ has been also tested in the experiments. However, the obtained results were not significant as *NGTh* on the accuracy.

4.4 Summary

Due to the distributed architecture of the SW, the quality of data is under investigation. This chapter concentrates on a SW data quality issue called ICA problem that indicates incorrect assignments between instance-level data and classes in the ontology. This chapter proposed an entropy-based method called CAD to deal with the ICA problem. The CAD evaluates the correctness and incorrectness of relationships between instance-level data and classes. The CAD has been tested over some classes of DBpedia dataset (3.8). The conducted experiments show the effectiveness and accuracy of the CAD. Additionally, the CAD is applicable to any kind of RDF-style KBs.

The work in this chapter has been published in (Barati, Bai & Liu, 2018).

Chapter 5

Automated New Classes Extraction From RDF Data

Ontology enrichment is one of the most significant tasks in the process of engineering and refining ontologies. An ontology has to go through a repetitive procedure of purification during its development life cycle. Namely, ontologies require automatic enrichments since they have not provided all requirements for users. In the SW, ontology enrichment is a research topic that is studied from various angles such as schema learning, relation learning, class learning, etc. Class learning is one of the most demanding aspects in this field of study aimed to enrich ontologies by extracting new classes. On this subject, most existing studies are built based on ILP techniques for deriving new classes from SW data. As discussed earlier, ILP-based methods are suitable for logical scenarios that usually require counterexamples.

To overcome with the limitation of counterexamples used in the ILP-based methods, this chapter proposes a non-logical approach called Class Enricher (CEn) that automatically extracts new classes with appropriate signatures through mining instance-level and schema-level knowledge.

The rest of this chapter is structured as follows. Section 5.1 illustrates the motivation used throughout of chapter. The foundation of the CEn approach is clearly described in Section 5.2. To check the ability of CEn approach in generating new classes, Section 5.3 discusses the experimental results. Finally, Section 5.4 summarises the whole chapter and reviews the contributions of the CEn approach.

5.1 Motivation

Based on the discussion in Chapter 4, a Common Feature or a Virtual Common Feature is a Class Feature if the information gained by the Common Feature or Virtual Common Feature is greater than or equal to NormGain Thresholds (*NGTh*). Consequently, Instance-Class Relationship Analysis Algorithm (Algorithm 4.2) in Chapter 4 evaluates the correctness and incorrectness of relationships between instances and classes.

However, a Common Feature or a Virtual Common Feature with information lower than *NGTh* can potentially be a signature for a new class; because the Common Feature or Virtual Common Feature indicates a behaviour that is frequently occurred by instances. This idea has been used throughout this chapter for generating new classes. Obviously, any instance with CA, ICA, or Undecidable status can share common features or virtual common features with information lower than *NGTh*. Consider Figure 5.1 to clarify the idea by providing an example. In the figure, (*genre, Pop*), (*influencedBy, Picasso*), (*influencedBy, Shakespeare*), and (*subject, Travel writing*) are features of MusicalArtist, Painter, Poet, and Writer classes, respectively. Now, suppose that (*like, Pizza*) is a Common Feature with information lower than *NGTh*. Although (*like, Pizza*) is not eligible to be used as a Class Feature, it can be exploited as a signature for a new class; because (*like, Pizza*) reveals a common behaviour among some instances of the MusicalArtist, Painter, and Writer classes. By considering Figure 5.1, a new class with the signature of (*like, Pizza*) contains 7 instances.

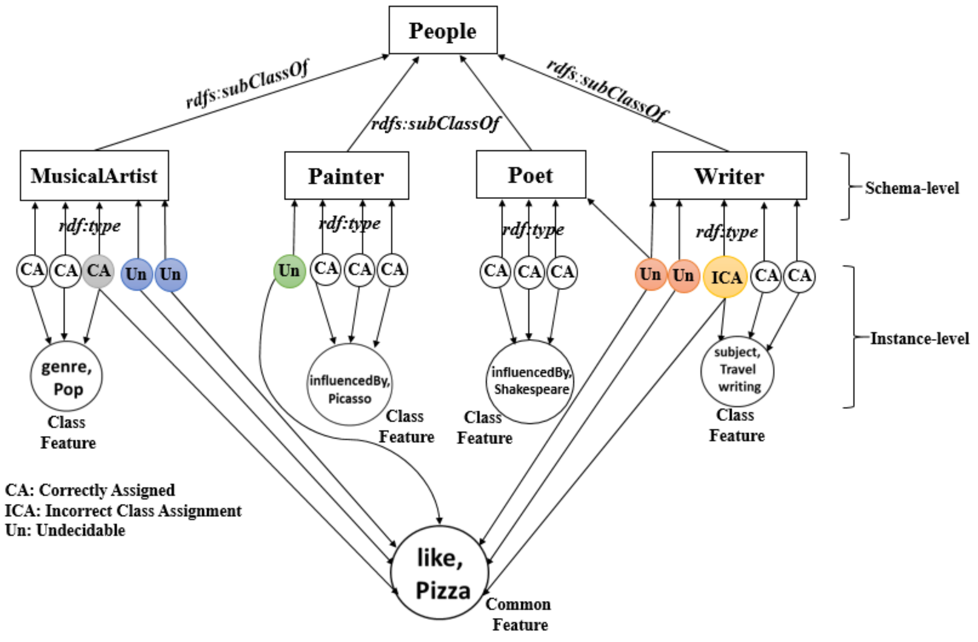

Figure 5.1: An example of instances that share a Common Feature with information lower than *NGTh*

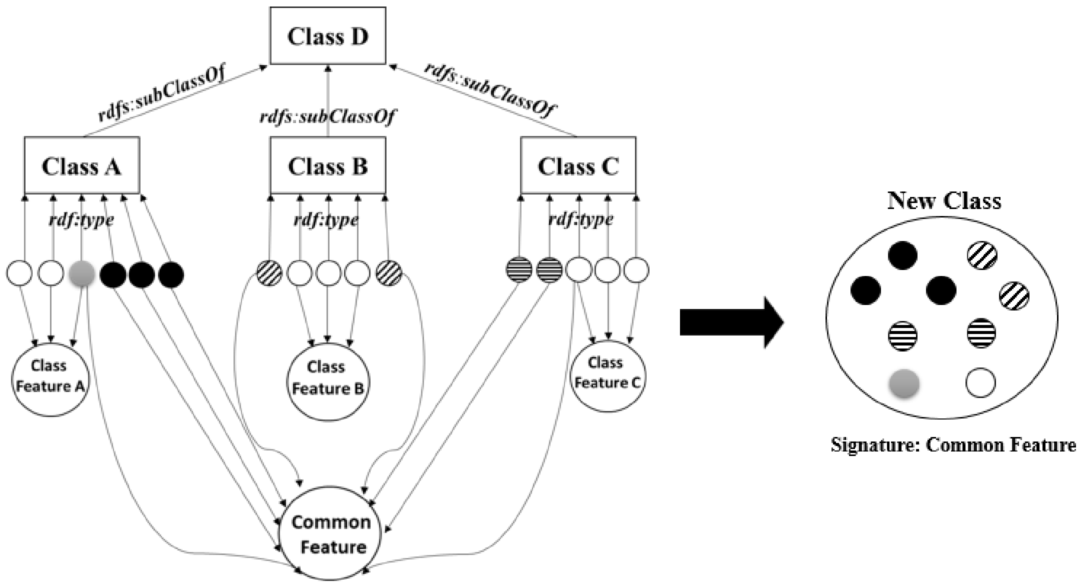

Figure 5.2: A new class created by a Common Feature

Figure 5.2 represents an example of a new class generated by a Common Feature that is used as a signature of the class. Obviously, the number of instances in the new classes should be greater than or equal to Minimum Instance Number (*MinIN*) that was introduced in Chapter 4.

5.1.1 New class placement

Any class has a particular position in the hierarchical structure of an ontology. In our scenario, a new generated Class c_i is a subclass of a predefined Class c_j in the ontology, where c_j is a lowest common class of all instances in c_i. The reason behind this idea is that instances of a subclass are members of its superclasses in the ontological structure.

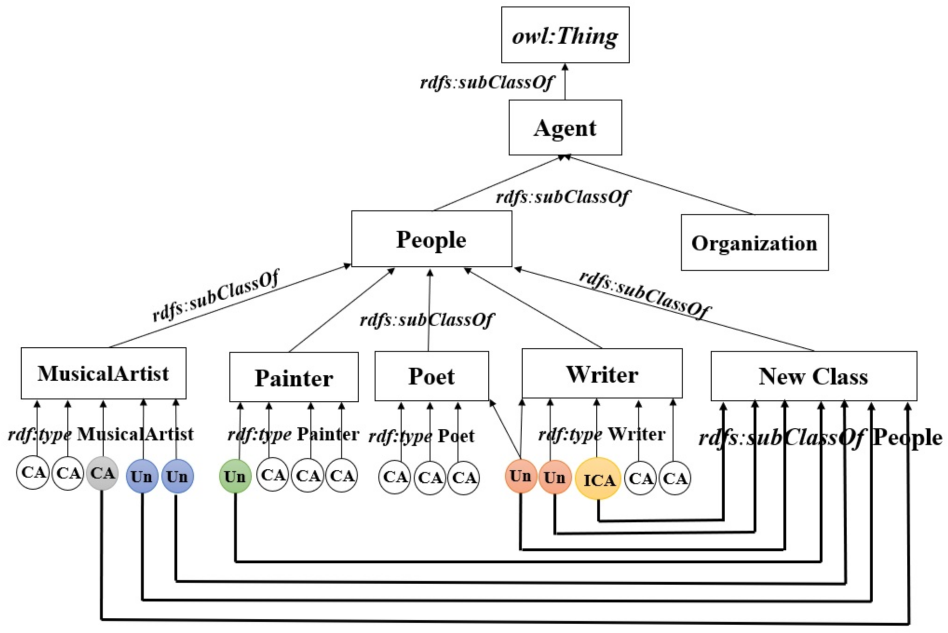

Figure 5.3: An example of new class placement

Consider again Figure 5.1 to explain where to put the new class in the hierarchical structure of an ontology. The new class with the signature of (*like, pizza*) is built by five Undecidable, one CA, and one ICA instances. As shown in Figure 5.3, the People class is the lowest common class for all instances in the new class. In the SW, instances can have multiple *rdf:type* properties. For example, there exists a pink Undecidable instance in Figure 5.1 that is a member of Writer and Poet classes. For this Undecidable instance with multiple types, the super class is People class. Accordingly, the new class is a subclass of People class and its place is parallel with MusicalArtist, Painter, Poet, and Writer classes in the hierarchical structure of the ontology. Figure 5.3 represents instances of the New Class in the enriched schema. These instances not only inherit the type of New Class that is *rdfs:subClassOf* People but also they share the types of predefined classes in the ontology.

5.2 Class Enricher (CEn) approach

The goal of the CEn approach is to enrich schema-level knowledge by extracting new classes which are not defined in the ontology. In this respect, the CEn approach contains two main modules: (I) Information Gain Analysis module and (II) Class Creation module. In general, the Information Gain Analysis module extracts instances that have shared common features or virtual common features with information lower than *NGTh*. Based on the results obtained by the Information Gain Analysis module, Class Creation module generates new classes to enrich schema-level knowledge.

5.2.1 Information Gain Analysis module

In the following, the Information Gain Analysis Algorithm is proposed that receives Knowledge Base (KB), $NGTh$, and instances ($Iset$) as inputs. Then, the algorithm generates $Inset_{cfs}$ and $Inset_{vcfs}$ that contain instances with common features or virtual common features with information lower than *NGTh*.

For each instance ins_i in *Iset*, Algorithm 5.1 checks if ins_i shares common features or virtual common features with information lower than *NGTh* (Lines 3-4). If the condition in Line 4 is true, then the algorithm records common features and virtual common features of ins_i in $Inset_{cfs}$ and $Inset_{vcfs}$ sets, respectively (Lines 5-11).

Algorithm 5.1: Information Gain Analysis

```
1: INPUT: $KB, NGTh, Iset$
2: OUTPUT: $Inset_{cfs}$, $Inset_{vcfs}$
3: for each $ins_i \in Iset$ do
4:    if $ins_i$ has (common features || virtual common features) && the information of each Common
      Feature or each Virtual Common Feature is lower than NGTh then
5:       if $ins_i$ shares common features then
6:          $Inset_{cfs} \leftarrow add \{ins_i$, common features$\}$;
7:       end if
8:       if $ins_i$ shares virtual common features then
9:          $Inset_{vcfs} \leftarrow add \{ins_i$, virtual common features$\}$;
10:      end if
11:   end if
12: end for
13: return $Inset_{cfs}$, $Inset_{vcfs}$;
```

5.2.2 Class Creation module

In this module, the Class Creation Algorithm is proposed to generate new classes by using the outputs collected by Algorithm 5.1. In the following, Algorithm 5.2 receives KB, $Inset_{cfs}$, $Inset_{vcfs}$, and $MinIN$ as inputs and generates new classes as output.

Algorithm 5.2: Class Creation

```
1: INPUT: $KB$, $Inset_{cfs}$, $Inset_{vcfs}$, $MinIN$;
2: OUTPUT: $generatedClasses$;
3: for each Common Feature or Virtual Common Feature of $\{ins_i \mid ins_i \in (Inset_{cfs} \parallel Inset_{vcfs})\}$
   do
4:    Create $class$ with signature of Common Feature or Virtual Common Feature of $ins_i$;
5:    $class \leftarrow add\{ins_j \mid ins_j \in (Inset_{cfs} \parallel Inset_{vcfs})$ && $ins_j$ has Common Feature or Virtual
      Common Feature of $ins_i\}$;
6:    if $\mid class \mid \geq MinIN$ then
7:       $Type \leftarrow LCS(\{ins_j \mid ins_j \in class\})$;
8:       $generatedClasses \leftarrow add\{class, Type\}$;
9:    end if
10: end for
11: return $generatedClasses$;
```

For each Common Feature or Virtual Common Feature of instance ins_i, Algorithm

5.2 creates new classes with the proper signatures (Lines 3-4). The algorithm identifies instances in the $Inset_{cfs}$ or $Inset_{vcfs}$ that have the same Common Feature or Virtual Common Feature with ins_i and adds them to the new $class$ (Line 5). If the number of instances in the new *class* is greater than or equal to *MinIN*, then the algorithm locates new class under the Lowest Common Superclass (*LCS*) in the hierarchical structure of the ontology (Lines 6-7). The algorithm adds new classes to *generatedClasses* set in Line 8.

Regarding to the output of Algorithm 5.2, it is important to mention that though a new class will increase the overall entropy (i.e., uncertainty) of Class random Space as the total number of classes increased, the new class will not impact on common feature spaces. Hence, the entropies of common features do not need to be re-calculated (refer to Definition 4.3.4).

5.3 Experiments and analysis

The goal of the following experiments is to check the effectiveness of CEn approach in generating new classes which are not defined in the DBpedia ontology. It is important to remind that in the following experiments, the CEn approach mines common features by using $\{s_1, s_2, ..., s_n\}$(p,o) structure.

5.3.1 Generating new classes for DBpedia ontology

Experimental set-up. The CEn approach has been tested over the dataset extracted from DBpedia 3.8 used in Chapter 4. The RDF dataset contains about 4800 triples built by 10 classes including Hotel, Food, ArchitecturalStructure, Museum, Beverage, Event, Holiday, BritishRoyalty, Person, and Writer. The new classes have been generated by considering *NGTh*<0.85 and *MinIN*≥15.

Experimental results. Table 5.1 shows some new classes along with their instances generated by the CEn approach. New classes have been assigned to suitable classes in the DBpedia ontology as noted in the table. As shown in Table 5.1, the CEn identifies Common Feature (*deathPlace, Beijing*) with information lower than *NGTh* as a signature of new Class c_2. This class contains some Chinese royalties, Chinese writers and Chinese people who were died in Beijing. As mentioned before, DBpedia ontology defines only two classes for royalties including BritishRoyalty and PolishKing. Table 4.4 in Chapter 4 also lists the features of BritishRoyalty class. Here, in Class c_2, all Chinese emperors have *rdf:type* BritishRoyalty property. Obviously, this kind of assignment does not make sense for human beings. In fact, this ambiguity is beacuse of lack of defining proper classes in the DBpedia ontology. On this subject, the CEn approach generates the new Class c_2 and assigns it to the People class that is the lowest common superclass class for all instances of c_2 in the DBpedia ontology.

Table 5.1: Some new classes generated by the CEn approach (*NGTh*<0.85, *MinIN*≥15)

Class	**class signature**	**NormGain**	**Instances**
c_1: *rdf:type* People	(*deathPlace, Germany*)	0.51	{*Paul Heyse, Maria Anna Sophia of Saxony, Theodor Fontane, Maximilian I Joseph of Bavaria, Charles Theodore Elector of Bavaria, Karl May, Charles VII Holy Roman Emperor, Maximilian II Emanuel Elector of Bavaria, Jean Paul, Princess Henriette Adelaide of Savoy, Gerhart Hauptmann, Maximilian I Elector of Bavaria, Friedrich Schiller, ...*}

c_2 *rdf:type* People	(*deathPlace*, *Beijing*)	0.45	{*Yongle Emperor, Mao Dun, Zhengde Emperor, Guo Moruo , Wanli Emperor, Chongzhen Emperor, Zhou Zuoren, Taichang Emperor, Xuande Emperor, Cao Xueqin, Longqing Emperor, Hongxi Emperor, Jiajing Emperor, Ai Qing, ...*}
c_3 *rdf:type* People	(*deathPlace*, *Portugal*)	0.41	{*Gil Vicente, John III of Portugal, Almeida Garrett, Afonso III of Portugal, Afonso I of Portugal, Camilo Castelo Branco, John I of Portugal, Antero de Quental, Manuel I of Portugal, Henry King of Portugal, Afonso V of Portugal, Alexandre Herculano, ...*}
c_4 *rdf:type* Food	(*origin*, *United States*)	0.6	{*Chicken fried bacon, Crystal Pepsi, Chili con carne, Coca-Cola Cherry, A&W Root Beer, AMP Energy, Pepsi Blue, Caffeine-Free Pepsi, A&W Cream Soda, Crunk Energy Drink, Caffeine-Free Coca-Cola, Cricket Cola, ...*}
c_5 *rdf:type* Food	(*origin*, *United Kingdom*)	0.43	{*Yorkshire pudding, Coca-Cola with Lemon, Spotted dick, Bedfordshire clanger, Relentless (drink), Steak and kidney pudding, Frijj, Treacle sponge pudding, Bread and butter pudding, Sticky toffee pudding, Qibla Cola, Suet pudding, Virgin Cola, Queen of Puddings, ...*}

Additionally, Table 4.4 lists features of the Food class and the Beverage class. Table 5.1 shows two new Classes c_4 and c_5 that collect popular food and beverage in the United States and the United Kingdom, respectively. The CEn approach assigns these classes to Food class that is the lowest common superclass for instances of c_4 and c_5 in the DBpedia ontology. More precisely, instances of c_4 and c_5 more likely indicate the United States and United Kingdom popular Food.

5.4 Summary

Ontologies can be incomplete since they have a quite static nature and the terminological axioms describing specific domains usually create before actual usage. A class is a particular notion that is known as one of the fundamental components in the ontological structures. Since OWL ontologies are developed by using DL languages, different studies are also designed based on DL-based techniques to learn new classes from the SW data to enrich ontologies. However, these methods usually need predefined logical knowledge to extract classes from instances. This chapter introduces a non-logical approach called CEn to creates new classes in the SW. The CEn approach automatically mines instance-level data by considering knowledge encoded at the schema-level to generate new classes that are not defined in the original ontology. This promised the potential of automated knowledge generation from the SW data.

Chapter 6

Conclusions

This chapter summarises the outputs of this thesis in mining and learning from the SW. I have highlighted the findings of this thesis in Section 6.1. The limitations of the proposed methods and possible future work are also explained in Section 6.2.

6.1 Research contributions

The goal of the SW is to convert the unstructured data on the Web pages into structured content. The SW provides tools to define a semantic layer on top of heterogeneous resources. A semantic approach to data processing, such as the use of ontologies and RDF-style KBs, has increasingly been integrated with the artificial intelligence techniques. These techniques improve the traditional approaches of information retrieval and data management and they result in more efficient and scalable information processing and easier human-machine interaction. In the SW, RDF-style KBs contain millions of triples that face with new challenges. Ontologies provide semantics for the RDF data, but the potential behind them have not effectively penetrated in the mining process. Furthermore, the KBs usually face with incompleteness and incorrectness issues due to the distributed nature of the SW.

In this thesis, the following three research questions have been identified and defined:

1. How to automatically mine semantic association rules from RDF triples by utilizing instance-level and schema-level knowledge?
2. How to analyse the correctness and incorrectness of relationships between instances and classes in the ontology?
3. How to automatically enrich schema-level knowledge by adding new classes which are not defined in the ontology?

Regarding to Research Question 1, I proposed the SWARM approach in Chapter 3. SWARM can automatically mine semantic association rules from different types of entities. SWARM has used instance-level data and schema-level knowledge for generating semantically-enriched rules. Regarding to Research Question 2, the CAD approach was proposed in Chapter 4 to assess the correctness and incorrectness of relationships between instances and classes in ontologies. To satisfy the objectives of Research Questions 3, the CEn approach was proposed in Chapter 5 to automatically enrich schema-level knowledge by identifying new classes which are not defined in the ontology.

This research contributes to the field in the following ways:

- **It shows the importance of utilising schema-level knowledge for attaching semantics to the discovered patterns**: The studies done in this thesis illustrates that ignoring ontological relations between instances and classes causes ambiguous interpretations. To mitigate this problem, this thesis proposes the SWARM that is a mining approach which reveals behavioural patterns from the SW data. The SWARM approach generates semantically-enriched association rules by exploiting schema-level potential beside of instance-level data. More precisely, it uses *rdf:type* and *rdfs:SubClassOf* properties to mine semantically-enriched rules. These properties have been used to extract class information of instances to measure rule quality factors. It is the main reason that makes the SWARM different from other mining methods which focus on discovering association rules in the SW. The experimental results indicate that the SWARM outperforms the existing methods by attaching more semantics to the discovered rules.

- **It tackles a SW data quality issue to improve the incorrectness and inconsistencies in the RDF-style KBs**: This thesis concentrates on a SW data quality issue called ICA that represents incorrect assignments between instances at the instance-level and classes in the ontology. This thesis proposes an approach called CAD to deal with the ICA problem. The CAD approach evaluates the correctness and incorrectness of relationships between instances and classes. In the CAD approach, the concept of traditional information theory has been borrowed and adapted to the SW data to extract features of classes in the ontology.

- **It boosts the process of ontology enrichment by discovering new classes**: Since ILP-based methods are suitable for logical applications and usually work with counterexamples, this thesis proposes an approach called CEn that groups instances with common attributes into specific classes to enrich ontologies. The CEn is a non-logical approach that discovers new classes without counterexamples and pre-defined logical knowledge. The CEn approach mines instance-level data by considering knowledge encoded at schema-level to create new classes which are not defined in the ontology. Additionally, the CEn approach puts generated classes in suitable places in the hierarchical structure of an ontology.

6.2 Future directions

The following discusses some possible improvements for the proposed methods and also further directions for future studies.

Firstly, the SWARM concentrates on mining semantic association rules from a single ontological structure such as DBpedia ontology. In the SW, RDF-style KBs are connected to each other. For example, DBpedia datasets not only provides information about its ontological structure but also about other structures such as YAGO. In the DBpedia datasets, most instances have been assigned to the YAGO ontology. For example, "*Black Swan*" is assigned to only two classes in the DBpedia ontology, while it is a member of more than thirty different classes in the YAGO ontology. It could be interesting to develop an approach to mine semantic association rules by using multiple structures in the SW. Another important direction for developing SWARM approach is to use other semantic properties encoded at the schema-level to generate semantically-enriched rules. The time complexity of the SWARM algorithm belongs to the O(n^2) class (including the time for generating the Semantic Items, Common Behaviour Sets, and Semantic Association Rules). In the future, it could be beneficial to improve the execution time of SWARM to deal with larger datasets.

Secondly, the CAD approach evaluates the correctness and incorrectness of *rdf:type* properties used between instances and classes. By considering an ontological structure, a future direction is to define features for super-classes by extracting the features of subclasses. To get a better understanding of the direction, consider a superclass along with its subclasses. By identifying the features of a superclass, it is possible to modify the *rdf:type* property of an instance if it has been incorrectly assigned to the subclass instead of the superclass. Equally interesting, another direction for future work is to use Natural Language Processing (NLP) techniques on features to find out more similar behaviours taken by instances. For example, consider group features gf_1=(like, cheese)$\{s_1, s_2, ..., s_n\}$, gf_2=(hate, cheese)$\{s_1, s_2, ..., s_n\}$, and gf_3=(love, cheese)$\{s_1, s_2, ..., s_n\}$. By employing NLP techniques on the features, it is possible to discover more interesting behavioural patterns since "like" and "love" are synonyms while "hate" is opposite of them.

Thirdly, the CEn approach generates new classes which are not defined in an ontology. An interesting future direction is to develop an intelligent approach to extract subclasses from new discovered classes. For this purpose, we need to formally define the principles for generating subclasses in the context of the SW. It could also be interesting to run CEn approach on larger RDF-style datasets to get a better understanding of its performance.

Besides specific improvements explained above, the future work could also involve in the domain and range enrichment. Given an RDF triple, i.e., (subject, predicate, object), the domain of a predicate states the class type of a subject while the range of a predicate indicates the class type of an object. Sometimes the domain and range properties determined in the ontology do not provide enough semantics for instance-level data. In the DBpedia datasets, the domains of many predicates defined on *owl:Thing* class that is a root class. For example, DBpedia ontology defined *Domain:owl:Thing* and *Range:Agent* for *foundedBy*[8] as a predicate. According to this knowledge encoded in the ontology, we can interpret that anything can be founded by an *Agent*. More precisely, any subclass of *owl:Thing* class could be considered as a domain for *foundedBy*. It is obvious that defining *owl:Thing* as a domain is too generic for *foundedBy*. It is also worth to know that this issue is not limited to the domain properties. Consider *Category*[9], *authority*[10], and *builder*[11] as predicates defined in the DBpedia ontology. The domain and range properties of these predicates are defined on *owl:Thing* class that

[8]http://dbpedia.org/ontology/foundedBy
[9]http://dbpedia.org/page/Category
[10]http://dbpedia.org/ontology/authority
[11]http://dbpedia.org/ontology/builder

reflects a super general concept. As seen, it is another quality issue that can be studied as a future direction for refining ontologies in the SW.

References

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

Abedjan, Z. & Naumann, F. (2013b). Synonym analysis for predicate expansion. In *Extended semantic web conference* (pp. 140–154).

Abedjan, Z. & Naumann, F. (2014). Amending rdf entities with new facts. In *The semantic web: Eswc 2014 satellite events* (pp. 131–143). Springer.

Acosta, M., Zaveri, A., Simperl, E., Kontokostas, D., Auer, S. & Lehmann, J. (2013). Crowdsourcing linked data quality assessment. In *International semantic web conference* (pp. 260–276).

Acosta, M., Zaveri, A., Simperl, E., Kontokostas, D., Flöck, F. & Lehmann, J. (2016). Detecting linked data quality issues via crowdsourcing: A dbpedia study. *Semantic Web*.

Agrawal, R., Imieliński, T. & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Acm sigmod record* (Vol. 22, pp. 207–216).

Ahmed, S. S., Malki, M. & Benslimane, S. M. (2015). Ontology partitioning: Clustering based approach. *International Journal of Information Technology and Computer Science (IJITCS)*.

Alexander, K. & Hausenblas, M. (2009). Describing linked datasets-on the design and usage of void, the'vocabulary of interlinked datasets. In *In linked data on the web workshop (ldow 09), in conjunction with 18th international world wide web conference (www 09*.

Amarilli, A., Galárraga, L., Preda, N. & Suchanek, F. M. (2014). Recent topics of research around the yago knowledge base. In *Web technologies and applications* (pp. 1–12). Springer.

An, Y. & Topaloglou, T. (2008). Maintaining semantic mappings between database schemas and ontologies. In *Semantic web, ontologies and databases* (pp. 138–152). Springer.

Antoniou, G. & Van Harmelen, F. (2004). *A semantic web primer*. MIT press.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. & Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *The semantic web* (pp. 722–735). Springer.

Bao, J., Kendall, E. F., McGuinness, D. L., Patel-Schneider, P. F., Ding, L. & Khandelwal, A. (2009). Owl 2 web ontology language quick reference guide. *World Wide Web Consortium, Recommendation REC-owl2-quick-reference-20091027*.

Barati, M., Bai, Q. & Liu, Q. (2016). Swarm: an approach for mining semantic association rules from semantic web data. In *Pacific rim international conference on artificial intelligence* (pp. 30–43).

Barati, M., Bai, Q. & Liu, Q. (2017). Mining semantic association rules from rdf data. *Knowledge-Based Systems*, *133*, 183–196.

Barati, M., Bai, Q. & Liu, Q. (2018). An entropy-based class assignment detection approach for rdf data. In *Pacific rim international conference on artificial intelligence* (pp. 412–420).

Basse, A., Gandon, F., Mirbel, I. & Lo, M. (2010). Dfs-based frequent graph pattern extraction to characterize the content of rdf triple stores. *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line, Raleigh, NC: US*.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001). The semantic web. *Scientific american*, *284*(5), 34–43.

Bichler, M. (2006). Design science in information systems research. *Wirtschaftsinformatik*, *48*(2), 133–135.

Bizer, C., Heath, T. & Berners-Lee, T. (2011). Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts* (pp. 205–227). IGI Global.

Böhm, C., de Melo, G., Naumann, F. & Weikum, G. (2012). Linda: distributed web-of-data-scale entity matching. In *Proceedings of the 21st acm international conference on information and knowledge management* (pp. 2104–2108).

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J. & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems* (pp. 2787–2795).

Broekstra, J., Kampman, A. & Van Harmelen, F. (2003). Sesame: An architecture for storing and querying rdf data and schema information. *Spinning the semantic web: Bringing the world wide web to its full potential*, *197*.

Bühmann, L. & Lehmann, J. (2013). Pattern based knowledge base enrichment. In *International semantic web conference* (pp. 33–48).

Bühmann, L., Lehmann, J. & Westphal, P. (2016). Dl-learner—a framework for inductive learning on the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, *39*, 15–24.

Cao, X., Zheng, Y., Shi, C., Li, J. & Wu, B. (2016). Link prediction in schema-rich heterogeneous information network. In *Pacific-asia conference on knowledge discovery and data mining* (pp. 449–460).

Chen, G., Wei, Q. & Kerre, E. E. (2000). Fuzzy data mining: Discovery of fuzzy generalized association rules+. In *Recent issues on fuzzy databases* (pp. 45–66). Springer.

Chi, Y., Muntz, R. R., Nijssen, S. & Kok, J. N. (2005). Frequent subtree mining–an overview. *Fundamenta Informaticae*, *66*(1-2), 161–198.

Chomboon, K., Kaoungku, N., Kerdprasop, K. & Kerdprasop, N. (2014). Data mining in semantic web data. *International Journal of Computer Theory and Engineering*, *6*(6), 472.

Consortium, W. W. W. et al. (2014). Rdf 1.1 concepts and abstract syntax.

d'Amato, C., Bryl, V. & Serafini, L. (2012). Semantic knowledge discovery from heterogeneous data sources. In *Knowledge engineering and knowledge management* (pp. 26–31). Springer.

d'Amato, C., Fanizzi, N. & Esposito, F. (2010). Inductive learning for the semantic web: What does it buy? *Semantic Web*, *1*(1, 2), 53–59.

d'Amato, C., Tettamanzi, A. G. & Minh, T. D. (2016). Evolutionary discovery of multi-relational association rules from ontological knowledge bases. In *European knowledge acquisition workshop* (pp. 113–128).

Debattista, J., Lange, C. & Auer, S. (2016). A preliminary investigation towards improving linked data quality using distance-based outlier detection. In *Joint international semantic technology conference* (pp. 116–124).

Debattista, J., Londoño, S., Lange, C. & Auer, S. (2015). Quality assessment of linked datasets using probabilistic approximation. In *European semantic web conference* (pp. 221–236).

Decker, S., Melnik, S., Van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., ... Horrocks, I. (2000). The semantic web: The roles of xml and rdf. *IEEE Internet computing*, *4*(5), 63–73.

Dehaspe, L. & Toivonen, H. (1999). Discovery of frequent datalog patterns. *Data Mining and knowledge discovery*, *3*(1), 7–36.

Dehaspe, L. & Toivonen, H. (2001). Discovery of relational association rules. In *Relational data mining* (pp. 189–212). Springer.

de Mantaras, R. L. & Saitia, L. (2004). Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text. In *16th european conference on artificial intelligence conference proceedings* (Vol. 110, p. 435).

De Nicola, A., Missikoff, M. & Navigli, R. (2009). A software engineering approach to ontology building. *Information systems*, *34*(2), 258–275.

Ding, L., Shinavier, J., Shangguan, Z. & McGuinness, D. L. (2010). Sameas networks and beyond: analyzing deployment status and implications of owl: sameas in linked data. In *The semantic web–iswc 2010* (pp. 145–160). Springer.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., ... Zhang, W. (2014). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th acm sigkdd international conference on knowledge discovery and data mining* (pp. 601–610).

Dongo, I., Cardinale, Y., Al-Khalil, F. & Chbeir, R. (2017). Semantic web datatype inference: towards better rdf matching. In *International conference on web information systems engineering* (pp. 57–74).

Dos Reis, J. C., Pruski, C. & Reynaud-Delaître, C. (2015). State-of-the-art on mapping maintenance and challenges towards a fully automatic approach. *Expert Systems with Applications*, *42*(3), 1465–1478.

Dou, D., Wang, H. & Liu, H. (2015). Semantic data mining: A survey of ontology-based approaches. In *Proceedings of the 9th ieee international conference on semantic computing* (pp. 244–251).

El-Sayed, M., Ruiz, C. & Rundensteiner, E. A. (2004). Fs-miner: efficient and incremental mining of frequent sequence patterns in web logs. In *Proceedings* *of the 6th annual acm international workshop on web information and data management* (pp. 128–135).

Esposito, F., Fanizzi, N. & d'Amato, C. (2007). Conceptual clustering applied to ontologies. In *International workshop on mining complex data* (pp. 42–56).

Fleischhacker, D., Paulheim, H., Bryl, V., Völker, J. & Bizer, C. (2014). Detecting errors in numerical linked data using cross-checked outlier detection. In *International semantic web conference* (pp. 357–372).

Fleischhacker, D. & Völker, J. (2011). Inductive learning of disjointness axioms. In *Otm confederated international conferences" on the move to meaningful internet systems"* (pp. 680–697).

Freitas, A., Curry, E., Oliveira, J. G. & O'Riain, S. (2012). Querying heterogeneous datasets on the linked data web: challenges, approaches, and trends. *IEEE Internet Computing*, *16*(1), 24–33.

Galárraga, L., Heitz, G., Murphy, K. & Suchanek, F. M. (2014). Canonicalizing open knowledge bases. In *Proceedings of the 23rd acm international conference on conference on information and knowledge management* (pp. 1679–1688).

Galárraga, L., Teflioudi, C., Hose, K. & Suchanek, F. M. (2015). Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, *24*(6), 707–730.

Galárraga, L. A., Preda, N. & Suchanek, F. M. (2013). Mining rules to align knowledge bases. In *Proceedings of the 2013 workshop on automated knowledge base construction* (pp. 43–48).

Galárraga, L. A., Teflioudi, C., Hose, K. & Suchanek, F. (2013). Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on world wide web* (pp. 413–422).

Gangemi, A., Nuzzolese, A. G., Presutti, V., Draicchio, F., Musetti, A. & Ciancarini, P. (2012). Automatic typing of dbpedia entities. In *International semantic web conference* (pp. 65–81).

Gao, W., Farahani, M. R., Aslam, A. & Hosamani, S. (2017). Distance learning techniques for ontology similarity measuring and ontology mapping. *Cluster Computing*, *20*(2), 959–968.

Géry, M. & Haddad, H. (2003). Evaluation of web usage mining approaches for user's next request prediction. In *Proceedings of the 5th acm international workshop on web information and data management* (pp. 74–81).

Giovanni, A., Gangemi, A., Presutti, V. & Ciancarini, P. (2012). Type inference through the analysis of wikipedia links. *Linked Data on the Web (LDOW)*.

Goczyła, K., Grabowska, T., Waloszek, W. & Zawadzki, M. (2006). The knowledge cartography–a new approach to reasoning over description logics ontologies. In *International conference on current trends in theory and practice of computer science* (pp. 293–302).

Goethals, B. & Van den Bussche, J. (2002). Relational association rules: Getting w armer. *Pattern Detection and Discovery*, 145–159.

Görlitz, O. & Staab, S. (2011). Splendid: Sparql endpoint federation exploiting void descriptions. In *Proceedings of the second international conference on consuming* *linked data-volume 782* (pp. 13–24).

Gunaratna, K., Thirunarayan, K., Sheth, A. & Cheng, G. (2016). Gleaning types for literals in rdf triples with application to entity summarization. In *International semantic web conference* (pp. 85–100).

Halpin, H., Hayes, P. J., McCusker, J. P., McGuinness, D. L. & Thompson, H. S. (2010). When owl: sameas isn't the same: An analysis of identity in linked data. In *International semantic web conference* (pp. 305–320).

Han, J., Cheng, H., Xin, D. & Yan, X. (2007). Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, *15*(1), 55–86.

Han, J., Pei, J. & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Acm sigmod record* (Vol. 29, pp. 1–12).

Hellmann, S., Lehmann, J. & Auer, S. (2009). Learning of owl class descriptions on very large knowledge bases. *International Journal on Semantic Web and Information Systems (IJSWIS)*, *5*(2), 25–48.

Horrocks, I. & Patel-Schneider, P. F. (2003). Three theses of representation in the semantic web. In *Proceedings of the 12th international conference on world wide web* (pp. 39–47).

Huan, J., Wang, W. & Prins, J. (2003). Efficient mining of frequent subgraphs in the presence of isomorphism. In *null* (p. 549).

Iannone, L., Palmisano, I. & Fanizzi, N. (2007). An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, *26*(2), 139–159.

Isele, R. & Bizer, C. (2012). Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, *5*(11), 1638–1649.

Isele, R., Jentzsch, A. & Bizer, C. (2011). Efficient multidimensional blocking for link discovery without losing recall. In *Webdb.*

Jirkovskỳ, V., Kadera, P. & Rychtyckyj, N. (2017). Semi-automatic ontology matching approach for integration of various data models in automotive. In *International conference on industrial applications of holonic and multi-agent systems* (pp. 53–65).

JÓzefowska, J., Ławrynowicz, A. & Łukaszewski, T. (2010). The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *Theory and Practice of Logic Programming*, *10*(3), 251–289.

Kasneci, G., Elbassuoni, S. & Weikum, G. (2009). Ming: mining informative entity relationship subgraphs. In *Proceedings of the 18th acm conference on information and knowledge management* (pp. 1653–1656).

Kaur, N. & Aggarwal, H. (2018). Query based approach for referrer field analysis of log data using web mining techniques for ontology improvement. *International Journal of Information Technology*, *10*(1), 99–110.

Kellou-Menouer, K. & Kedad, Z. (2015). Discovering types in rdf datasets. In *European semantic web conference* (pp. 77–81).

Khattak, A. M., Pervez, Z., Latif, K. & Lee, S. (2012). Time efficient reconciliation of mappings in dynamic web ontologies. *Knowledge-Based Systems*, *35*, 369–374.

Krompaß, D., Baier, S. & Tresp, V. (2015). Type-constrained representation learning in knowledge graphs. In *International semantic web conference* (pp. 640–655).

Kuramochi, M. & Karypis, G. (2001). Frequent subgraph discovery. In *Data mining, 2001. icdm 2001, proceedings ieee international conference on* (pp. 313–320).

Lao, N. & Cohen, W. W. (2010). Relational retrieval using a combination of path-constrained random walks. *Machine learning*, *81*(1), 53–67.

Lao, N., Mitchell, T. & Cohen, W. W. (2011). Random walk inference and learning in a large scale knowledge base. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 529–539).

Lehmann, J. (2009). Dl-learner: learning concepts in description logics. *Journal of Machine Learning Research*, *10*(Nov), 2639–2642.

Lehmann, J., Gerber, D., Morsey, M. & Ngomo, A.-C. N. (2012). Defacto-deep fact validation. In *International semantic web conference* (pp. 312–327).

Li, H. & Sima, Q. (2015). Parallel mining of owl 2 el ontology from large linked datasets. *Knowledge-Based Systems*, *84*, 10–17.

Lisi, F. A. & Esposito, F. (2005). Mining the semantic web: a logic-based methodology. In *Foundations of intelligent systems* (pp. 102–111). Springer.

Maedche, A. & Zacharias, V. (2002). Clustering ontology-based metadata in the semantic web. In *European conference on principles of data mining and knowledge discovery* (pp. 348–360).

McGuinness, D. L., Van Harmelen, F. et al. (2004). Owl web ontology language overview. *W3C recommendation*, *10*(10), 2004.

Melo, A., Völker, J. & Paulheim, H. (2017). Type prediction in noisy rdf knowledge bases using hierarchical multilabel classification with graph and latent features. *International Journal on Artificial Intelligence Tools*, *26*(02), 1760011.

Motik, B., Sattler, U. & Studer, R. (2005). Query answering for owl-dl with rules. *Web Semantics: Science, Services and Agents on the World Wide Web*, *3*(1), 41–60.

Muggleton, S. (1995). Inverse entailment and progol. *New generation computing*, *13*(3-4), 245–286.

Muggleton, S. & De Raedt, L. (1994). Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, *19*, 629–679.

Nebot, V. & Berlanga, R. (2012). Finding association rules in semantic web data. *Knowledge-Based Systems*, *25*(1), 51–62.

Nickel, M., Tresp, V. & Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (icml-11)* (pp. 809–816).

Parundekar, R., Knoblock, C. A. & Ambite, J. L. (2010). Linking and building ontologies of linked data. In *The semantic web–iswc 2010* (pp. 598–614). Springer.

Paulheim, H. (2014). Identifying wrong links between datasets by multi-dimensional outlier detection. In *Wodoom* (pp. 27–38).

Paulheim, H. & Bizer, C. (2013). Type inference on noisy rdf data. In *International semantic web conference* (pp. 510–525).

Peng, P., Zou, L., Özsu, M. T. & Zhao, D. (2018). Multi-query optimization in federated rdf systems. In *International conference on database systems for advanced applications* (pp. 745–765).

Quilitz, B. & Leser, U. (2008). Querying distributed rdf data sources with sparql. In *European semantic web conference* (pp. 524–538).

Ramakrishnan, C., Milnor, W. H., Perry, M. & Sheth, A. P. (2005). Discovering informative connection subgraphs in multi-relational graphs. *ACM SIGKDD Explorations Newsletter*, *7*(2), 56–63.

Raphaeli, O., Goldstein, A. & Fink, L. (2017). Analyzing online consumer behavior in mobile and pc devices: A novel web usage mining approach. *Electronic Commerce Research and Applications*, *26*, 1–12.

Rizzo, G., d'Amato, C., Fanizzi, N. & Esposito, F. (2016). Approximating numeric role fillers via predictive clustering trees for knowledge base enrichment in the web of data. In *International conference on discovery science* (pp. 101–117).

Rosemann, M. & Vessey, I. (2008). Toward improving the relevance of information systems research to practice: the role of applicability checks. *Mis Quarterly*, 1–22.

Ruckhaus, E., Baldizán, O. & Vidal, M.-E. (2013). Analyzing linked data quality with liquate. In *Otm confederated international conferences" on the move to meaningful internet systems"* (pp. 629–638).

Saleem, M., Ali, M. I., Hogan, A., Mehmood, Q. & Ngomo, A.-C. N. (2015). Lsq: the linked sparql queries dataset. In *International semantic web conference* (pp. 261–269).

Saleem, M. & Ngomo, A.-C. N. (2014). Hibiscus: Hypergraph-based source selection for sparql endpoint federation. In *European semantic web conference* (pp. 176–191).

Saruladha, K., Aghila, G. & Sathiya, B. (2012). A partitioning algorithm for large scale ontologies. In *Recent trends in information technology (icrtit), 2012 international conference on* (pp. 526–530).

Schmachtenberg, M., Bizer, C. & Paulheim, H. (2014). Adoption of the linked data best practices in different topical domains. In *International semantic web conference* (pp. 245–260).

Schneider, M., Carroll, J., Herman, I. & Patel-Schneider, P. F. (2009). Owl 2 web ontology language rdf-based semantics. *W3C Recommendation (October 27 2009)*.

Schwarte, A., Haase, P., Hose, K., Schenkel, R. & Schmidt, M. (2011). Fedx: Optimization techniques for federated query processing on linked data. In *International semantic web conference* (pp. 601–616).

Shadbolt, N., Berners-Lee, T. & Hall, W. (2006). The semantic web revisited. *IEEE intelligent systems*, *21*(3), 96–101.

Shah, R. & Jain, S. (2014). Ontology-based information extraction: An overview and a study of different approaches. *International Journal of Computer Applications*, *87*(4).

Shannon, C. E. (1949). Communication theory of secrecy systems. *Bell system technical journal*, *28*(4), 656–715.

Shanthi, S. (2017). Survey on web usage mining using association rule mining.

International Journal of Innovative Computer Science & Engineering, *4*(3), 65–67.

Singh, A. K., Kumar, A. & Maurya, A. K. (2014). Association rule mining for web usage data to improve websites. In *Advances in engineering and technology research (icaetr), 2014 international conference on* (pp. 1–6).

Sleeman, J. & Finin, T. (2013). Type prediction for efficient coreference resolution in heterogeneous semantic graphs. In *Semantic computing (icsc), 2013 ieee seventh international conference on* (pp. 78–85).

Spahiu, B., Xie, C., Rula, A., Maurino, A. & Cai, H. (2016). Profiling similarity links in linked open data. In *Data engineering workshops (icdew), 2016 ieee 32nd international conference on* (pp. 103–108).

Srikant, R. & Agrawal, R. (1995). *Mining generalized association rules.* IBM Research Division.

Suchanek, F. M., Abiteboul, S. & Senellart, P. (2011). Paris: Probabilistic alignment of relations, instances, and schema. *Proceedings of the VLDB Endowment*, *5*(3), 157–168.

Suchanek, F. M., Kasneci, G. & Weikum, G. (2007). Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on world wide web* (pp. 697–706).

Suh, S. C. & Gaddam, L. (2011). Role of clustering of ontology relations for preventive health care through nutrition. In *Mobile it convergence (icmic), 2011 international conference on* (pp. 126–132).

Sydow, M., Pikuła, M. & Schenkel, R. (2013). The notion of diversity in graphical entity summarisation on semantic knowledge graphs. *Journal of Intelligent Information Systems*, *41*(2), 109–149.

Takama, Y. & Hattori, S. (2007). Mining association rules for adaptive search engine based on rdf technology. *IEEE Transactions on Industrial Electronics*, *54*(2), 790–796.

Töpper, G., Knuth, M. & Sack, H. (2012). Dbpedia ontology enrichment for inconsistency detection. In *Proceedings of the 8th international conference on semantic systems* (pp. 33–40).

Tran, M. D., d'Amato, C., Nguyen, B. T. & Tettamanzi, A. G. (2017). An evolutionary algorithm for discovering multi-relational association rules in the semantic web. In *Proceedings of the genetic and evolutionary computation conference* (pp. 513–520).

Tran, M. D., d'Amato, C., Nguyen, B. T. & Tettamanzi, A. G. (2018). Comparing rule evaluation metrics for the evolutionary discovery of multi-relational association rules in the semantic web. In *European conference on genetic programming* (pp. 289–305).

Tsay, L.-S., Sukumar, S. R. & Roberts, L. W. (2015). Scalable association rule mining with predication on semantic representations of data. In *Technologies and applications of artificial intelligence (taai), 2015 conference on* (pp. 180–186).

Tseng, M.-C., Lin, W.-Y. & Jeng, R. (2008). Updating generalized association rules with evolving taxonomies. *Applied Intelligence*, *29*(3), 306–320.

Tsur, D., Ullman, J. D., Abiteboul, S., Clifton, C., Motwani, R., Nestorov, S. & Rosenthal, A. (1998). Query flocks: A generalization of association-rule mining. In *Acm sigmod record* (Vol. 27, pp. 1–12).

Tulasi, R. L. & Rao, M. S. (2014). Survey on techniques for ontology interoperability in semantic web. *Global Journal of Computer Science and Technology*, *14*(2).

Völker, J. & Niepert, M. (2011). Statistical schema induction. In *The semantic web: Research and applications* (pp. 124–138). Springer.

Volz, J., Bizer, C., Gaedke, M. & Kobilarov, G. (2009). Silk-a link discovery framework for the web of data. *LDOW*, *538*.

Wei, Q. & Chen, G. (1999). Mining generalized association rules with fuzzy taxonomic structures. In *Fuzzy information processing society, 1999. nafips. 18th international conference of the north american* (pp. 477–481).

Wen, Y., Jin, Y. & Yuan, X. (2018). Kat: Keywords-to-sparql translation over rdf graphs. In *International conference on database systems for advanced applications* (pp. 802–810).

Yaghouti, N., Kahani, M. & Behkamal, B. (2015). A metric-driven approach for interlinking assessment of rdf graphs. In *Computer science and software engineering (csse), 2015 international symposium on* (pp. 1–8).

Yan, X., Yu, P. S. & Han, J. (2004). Graph indexing: a frequent structure-based approach. In *Proceedings of the 2004 acm sigmod international conference on management of data* (pp. 335–346).

Yang, B., Yih, W.-t., He, X., Gao, J. & Deng, L. (2014). Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Yang, L. (2005). Pruning and visualizing generalized association rules in parallel coordinates. *IEEE Transactions on Knowledge and Data Engineering*, *17*(1), 60–70.

Yin, C., Gu, J. & Hou, Z. (2016). An ontology mapping approach based on classification with word and context similarity. In *Semantics, knowledge and grids (skg), 2016 12th international conference on* (pp. 69–75).

Zhang, D., Yang, Z., Wang, N., Wang, B. & Zhao, H. (2016). Ontology extension based on axiomatic cognitive model for ontology learning. In *Computer and communications (iccc), 2016 2nd ieee international conference on* (pp. 825–829).

Zhang, X., Zhao, C., Wang, P. & Zhou, F. (2012). Mining link patterns in linked data. In *Web-age information management* (pp. 83–94). Springer.

Zhu, M., Gao, Z., Pan, J. Z., Zhao, Y., Xu, Y. & Quan, Z. (2013). Ontology learning from incomplete semantic web data by belnet. In *Proceedings of 25th ieee international conference on tools with artificial intelligence* (pp. 761–768).

Appendix A

Glossary

SW Semantic Web

RDF Resource Description Framework

LOD Linked Open Data

XML eXtensible Markup Language

Turtle Terse RDF Triple Language

OWL Web Ontology Language

YAGO Yet Another Great Ontology

KBs Knowledge Bases

RDF/S Resource Description Framework Schema

URI Uniform Resource Identifier

SPARQL SPARQL Protocol and RDF Query Language

ILP Inductive Programming Language

SWRL Semantic Web Rule Language

DFS Depth-First Search

SWARM Semantic Web Association Rule Mining

SI Semantic Items

LLC Lowest Level Class

TCBS Total Common Behaviour Set

SD Similarity Degree

SimTh Similarity Thresholds

ICA Incorrect Class Assignment

CAD Class Assignment Detector

MinIN Minimum Instance Number

NGTh NormGain Thresholds

CA Correctly Assigned

ICA Incorrectly Assigned

Un Undecidable

CEn Class Enricher

NLP Natural Language Processing